An investigation of computerized information storage and retrieval methods, in a film library organized according to the universal decimal classification

Thesis

"An investigation of computerized information storage and retrieval methods, in a Film Library organized according to the Universal Decimal Classification."

by

Brian Eric Staff BSc., MSc.

Faculty of Mathematics
The Open University

October 1978

Submitted as a requirement for the degree of

Doctor of Philosophy

in the discipline of Computer Science

CONTENTS

ACKNOWLEDGEMENTS

Whilst working on this thesis I was financially supported by the Open University, the British Broadcasting Corporation and Sperry Univac, to whose generousity I owe a debt of gratitude.

For their tolerance of my ignorance, and forebearance of the work that I imposed upon them, I thank the personnel at the BBC film Library and, in particular, Kay Salway and Adrian Evans.

For encouragement and guidance I am indebted to Dr. P.G. Thomas, who took over supervision of the project at a late stage, but has since directed it with great insight.

To the many others who assisted me at one time or another (typists, computer staff etc.) I offer my warmest thanks, and for the punch unit at the Open University I would like to include my admiration for the quality and speed of their work.

Brian Staff
Open University
August 1978

SUMMARY

The operation of the BBC Film Library was studied with the intention of defining those areas likely to benefit from computerization. The state of the art of computerized information retrieval was assessed by means of the literature, and those techniques likely to be of use at the Film Library were isolated.

Computer programs were written to provide an information storage and retrieval system paralleling the manual system currently used at the Film Library, organized according to the UNIVERSAL DECIMAL CLASSIFICATION (UDC). These programs were operated by the film librarians in situ.

A computerized system able to "learn" from enquiries was built and tested, and document clustering was also investigated as a method of subject classification.

A modular approach to retrieval system design was developed within the framework of a Relational Database system, so that the various retrieval methods examined in the course of the study could be cemented into one concertive retrieval system.

Structure of this thesis

A brief INTRODUCTION is followed by CHAPTER 1, which introduces the reader to the Film Library, and CHAPTER 2 extends this acquantanceship by describing a statistical survey carried out there to assess the performance of the manual system. CHAPTER 3 is a literature survey covering library automation in general, and UDC computerization in particular. In CHAPTER 4, the various methods are discussed that might be of use at the Film Library, largely as intimated by the literature. In CHAPTER 5, a computer system based on the existing manual UDC system is described, and its use by Film Librarians is discussed. CHAPTER 6 describes a retrieval system capable of "learning" from queries, and CHAPTER 7 concerns experiments based on document clustering. In CHAPTER 8, a modular approach to retrieval system design is outlined, and put forward as a method of producing coherent, multi-faceted systems from individual building blocks. CHAPTER 9 briefly summarizes the important points to emerge from the project.

APPENDIX 1 and APPENDIX 2 are User Guides to the operation of the UDC-based retrieval and updating packages discussed in CHAPTER 5. APPENDIX 3 is a paper (at present under submission for publication) demonstrating a building block approach to retrieval system design after the Relational Database model, as shown in CHAPTER 8.

Introduction

The theme at the core of this thesis is the computerization of a film library. Many aspects of this area of information retrieval have been thoroughly dealt with by previous workers, who have applied simple algorithms to their problems and achieved gratifying results. Part of this thesis proceeds along similar lines, purely to avoid omission of alternatives that being simple, should be in no way be considered inferior.

Having covered the more obvious approaches however, less conventional systems were investigated, and finally a means of combining conventional and novel systems was considered, in the hope that such a unit would exceed the sum of the parts.

The time honoured library subject classifications (Dewey, UDC etc.) are respected as reliable and proven solutions to an intellectual problem. Modern computer methods that reject subjectively imposed subject structures fly the flag of automation, but currently lack the dependability of their manual counterparts. By proposing the development of computerized retrieval systems incorporating a combination of old and new techniques, one envisages the "push-me-

pull-you" progress of such diverse methods towards a solid and yet ambitious future.

The Open University's UNIVAC 1121 computer in the Data Processing Department was used for all of the programs described herein, with the exception of those mentioned in Chapter 8 and Appendix 3, which were run on the Hewlett-Packard 2000F belonging to the Student Computing Service of the Open University. The complete listings of all the programs written during the course of the project are lodged with my supervisor:-

Dr. P.G. Thomas

Maths Faculty

Open University

Milton Keynes

CHAPTER 1

Introduction to the Film Library

The BBC Film Library is situated at Brentford in Middlesex, and is the repository of all film footage shot by BBC camera crews, be it news items from the Lebanon, or drama inserts from "Z cars". The purpose of the Film Library is:-

(1) to physically store the reels of film, and

(2) to make the film accessible for re-use.

Requests for film items come from the various BBC production departments, and although any type of request is possible, it will generally fall into one of the following categories:-

(i) A programme is requested by title & transmission date, eg:-

"PANORAMA 12/2/77 on Rhodesia"

ie. by TITLE

(ii) Film on a specific subject, eg:-

"A Jumbo Jet in flight"

ie. by SUBJECT

(iii) Film of a well-known personality, eg:-

"Jim Callaghan outside number 10"

ie. by NAME

Requests vary greatly in their degree of specificity, and one is not always able to consider them as being of a distinct type, eg:-

"An item from PANORAMA discussing Jim Callaghan as he boards a Jumbo Jet in Rhodesia".

The Film Library is therefore committed to maintaining a number of access paths to the information that it holds, and this is achieved by the use of a set of files (each of which is tailored to a specific method of access) in order to satisfy the majority of requests.

The TITLE and NAME catalogues are 'non-intellectually' organized files in that they are ordered alphabetically, and thus constitute only the most elementary of information retrieval problems. In much of this thesis therefore, emphasis will be laid firmly on SUBJECT retrieval, although it is important that the power of the other files should not be overlooked.

Storage of information by SUBJECT is made possible by the use of a library organization scheme known as the

UNIVERSAL DECIMAL CLASSIFICATION

or UDC, which is therefore of central importance to the operational effectiveness of the Film Library.

Before going on to break down the information storage and retrieval process into its basic components, I think it worthwhile to look at the manner in which requests can vary, and the way in which this variation effects the structure of the system:-

(1) When a piece of film is stored away at the Film Library, the Librarians must avoid making assumptions as to the type of request that will lead to its retrieval at some future date. A comedy programme has as much right to a piece of political footage as a more seriously inclined programme, although they will doubtless take differing views of the same material. Ideally therefore, one would seek to avoid bias at the storage stage.

(2) Requests take greatly differing forms; from the rapid need for material on a suddenly newsworthy subject, to the more leisurely researched programmes made well in advance of transmission date.

(3) Requests are all too often constrained by the user's prejudgements. In search of a particular shot, a user will often rack his brains to remember the programme in which he saw a shot - thus replacing a SUBJECT request by a TITLE request even before contacting the Film Library.

Such problems as these are not of course unique, but any film library is greatly prone to the variation of interpretation that can be applied to the moving picture. The artistic temperament of creative programme makers will doubtless result in a different genre of requests to that coming from current affairs departments, and given a similar diversity of film output, one can easily see that any storage and retrieval system must be blessed with a sensitivity to the way in which the BBC operates.

The present system - an overall view

A schematic presentation of the storage system (by Subject) is given below:-

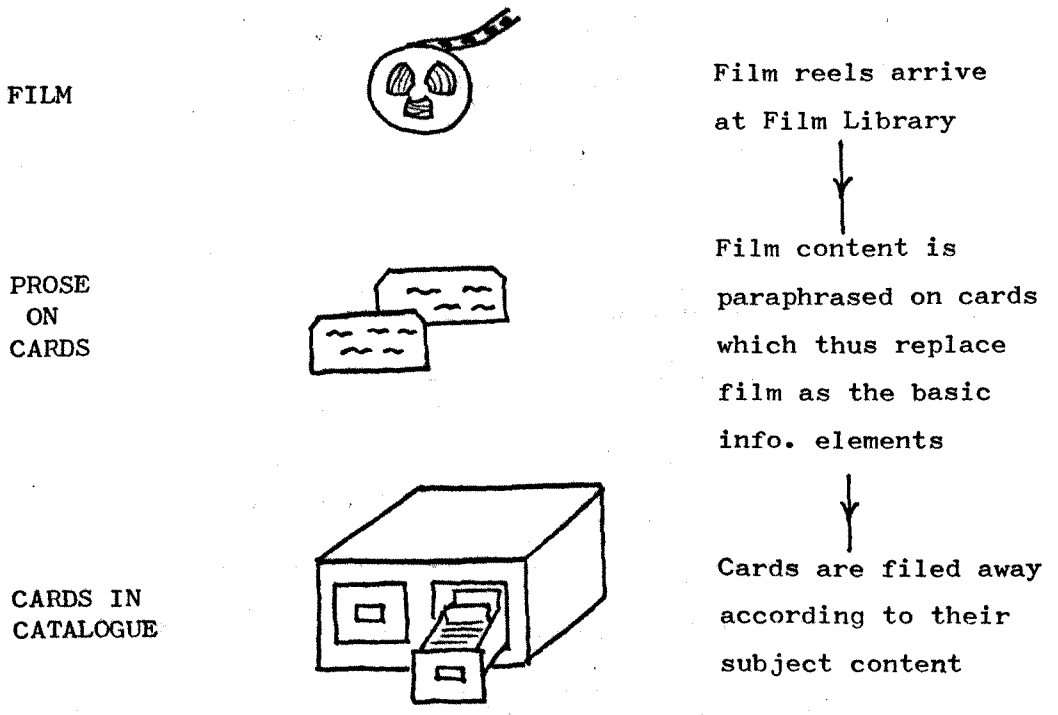


figure 1.1 - Overall Storage System

To be realistic, one can confidently say that certain components of the storage process will be disruptive, and, in a film library, none more so than the first stage, at which the very medium of the information is altered from film to text. To criticise this translation however, is to assume that there is some ready alternative, and this, sadly, is not the case. The prose description of the film, no matter how carefully composed, condenses the film imagery to an immeasurable degree, when to reduce a single frame (let alone a complete piece of action) to between 100 and 200 words, is something that can only be done drastically. Here, as in so many areas of the Film Library, shrewd anticipation of user requirements is vital in making the best of limited resources.

Following the reduction of film to prose, comes the point at which the item must be filed according to its subject matter (Note: the prose now is the item, the film being relegated to the vaults). This step, discussion of which forms the body of my thesis, concerns the derivation of a mapping that, despite the highly imperfect stage that preceeds it, must still be performed with maximum efficiency.

Now let us consider the basic retrieval process:-

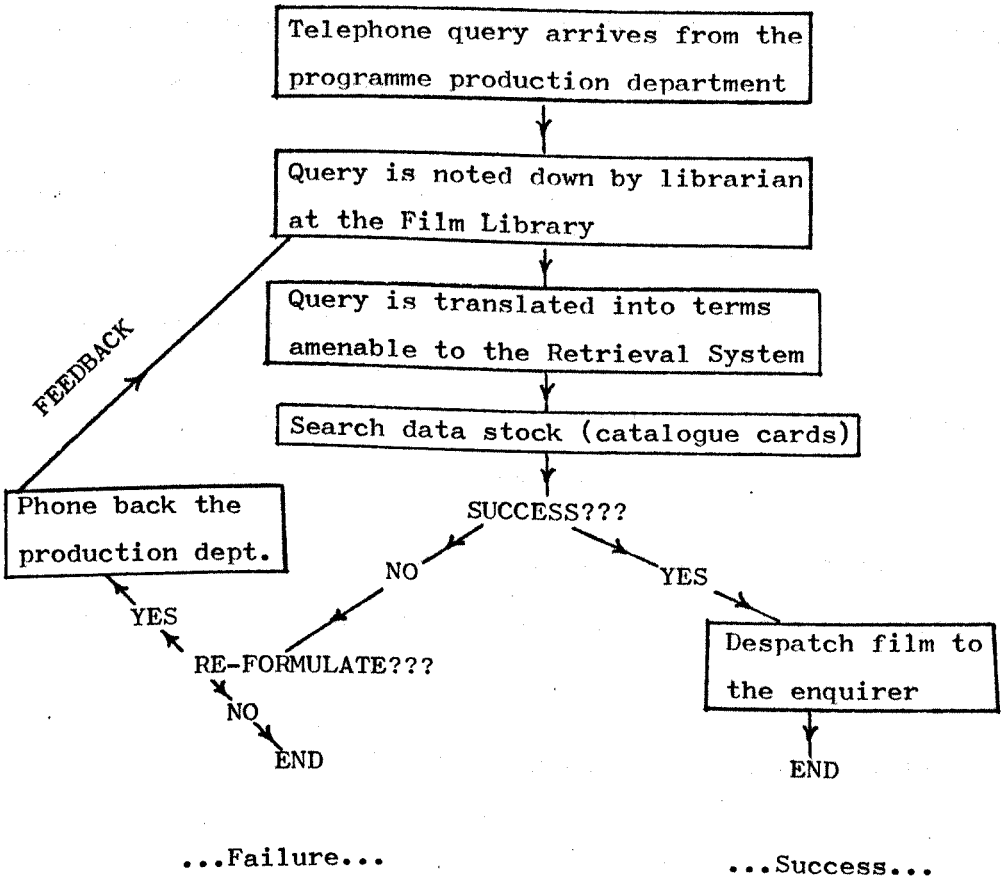


figure 1.2 - Overall Retrieval System

At first sight, it may appear that the feedback loop should be more immediate, for if the enquirer could react to the search as it takes place (rather than by a returned phone call), then perhaps some useful interaction could be established. Also, one questions whether the query should be forced into the mould that the system requires, or might not something of the opposite be possible?

These are vague criticisms, and levelled so generally that they apply to a whole range of retrieval systems. Specific to the Film Library however, is the mediation of a trained librarian between the enquirer and the system itself. Depending upon your point of view (and it may be as arbitrary a choice as that) this interposition of a personality can be regarded either as a potential source of confusion or as a beneficial mediation. [It has been suggested that the presence of an intermediate librarian can be justified solely because of the complex copyright restrictions, gauge sizes, soundtrack details etc., that attach to the filmic medium].

There are several important points to observe concerning the overall operation of the Film Library, which are really related to BBC rules of practice:

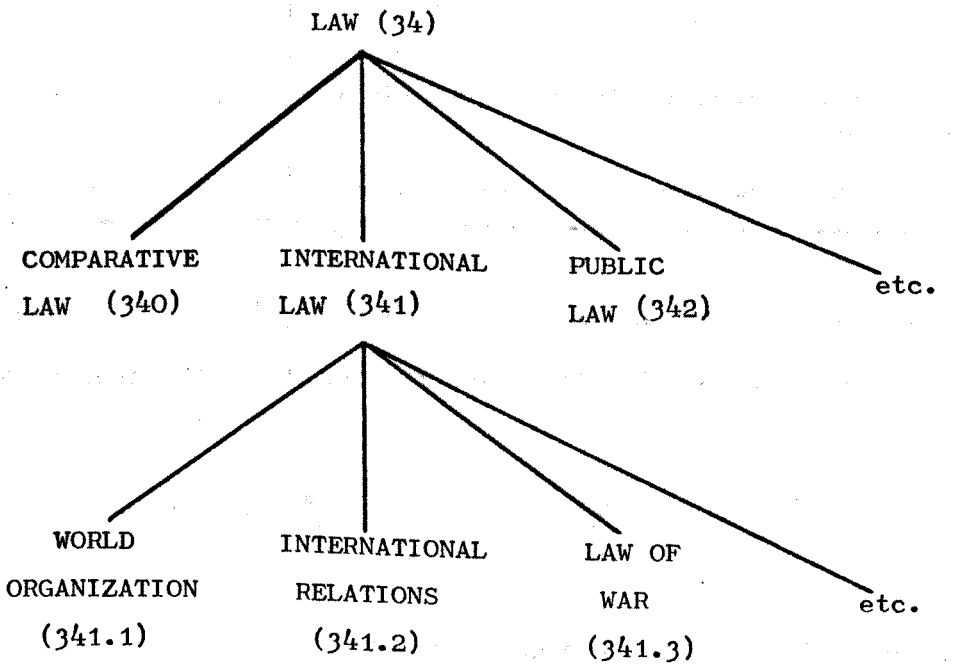
(1) Information Retrieval must be possible at all times, that is, the retrieval system cannot be prone to chronic malfunctions.

(2) Information Storage is not supported with such urgency, due both to accession delays (caused largely by the production departments themselves) and subject classification workload.

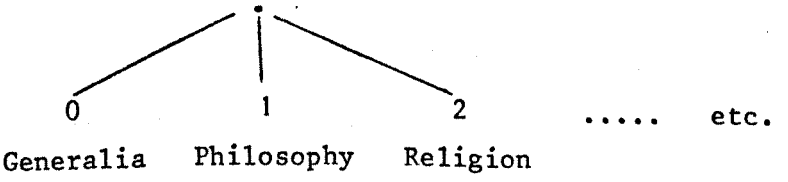
(3) Selected methods of Retrieval access are possible (reflecting storage policy), covering the majority of user requests. The switching of methods (Subject to Title, Title to Name and so on) may be undertaken by the librarian to salvage retrieval failures.

The UDC - a brief description

The UDC comprises a method of replacing concepts expressed in words by (mnemonic) codes. For example, the concepts LAW & TERRORISM are replaced by predefined UDC codes 34 & 343.77 respectively. For this purpose, the whole realm of human experience and abstraction is broken down hierarchically, eg:-



Being based on a decimal number system, a maximum 10 way splitting at each node is possible, including the very root of the classification thus:-



The decimal point is embedded in numbers to increase legibility only. Any single subject can therefore be expressed in terms of a string of digits and points. To denote multiple subjects however, it is necessary to use relational connectors, such as the colon, eg:-

Football 796.332

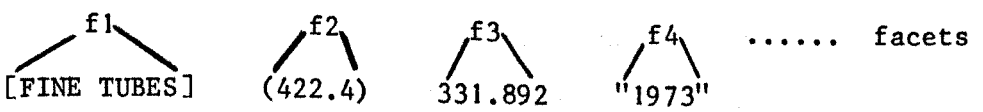
Crowds 301.182

Football Crowds 796.332:301.182

The UDC also contains certain devices to denote particular components, eg:-

DEVICE	MEANING	EXAMPLE
"..."	Time	"1942" refers to the year 1942 AD
[...]	Companies	[BBC] refers to the British Broadcasting Corporation
(...)	Place	(42) refers to the United Kingdom

So, a UDC string can be made up of a number of facets, where a "facet" may be an individual UDC code, or one of the above components, eg:-



Also, there are substrings that have special meaning. The substring .007 for example, signifies people, and this "auxiliary" as it is known, can be attached to other numbers to change their meaning, eg:-

621 = engineering

621.007 = engineers

These substrings are position independent, that is, their meaning is fixed no matter where they appear. This is not the case for other digital elements within UDC strings. For example, in 341.123 and 621.123, the .123 group has no common meaning - it simply depends on the organization in the LAW and SCIENCE areas respectively .

This is a simplified description of the UDC, and in practice one encounters the most horrendous strings in which yet more devices are used, eg:-

[CUBITTS] 331.892:69.007.25 (421.5 THAMESMEAD)"1969"

(In the above example, "name extension" is used to increase the specificity of the subject definition independently of the hierarchy, thus permitting THAMESMEAD to be directly referenced).

For a complete definition of the UDC, the reader should refer to the relevant British Standard [234], and for a more detailed explanation to FOSKETT [235], PERREAULT [236] and MILLS [237]. However, in addition to the standard UDC definition to be found in the above works, any individual implementation of the scheme is likely to contain some special features to suit its own purposes, and this is particularly true of media libraries, in which the medium itself demands expression. At the BBC Film Library, two additional facet types are used:-

R denotes camera motion shots, eg. R13 = Air to Ground

M denotes vehicular motion, eg. M65 = Aircraft landing

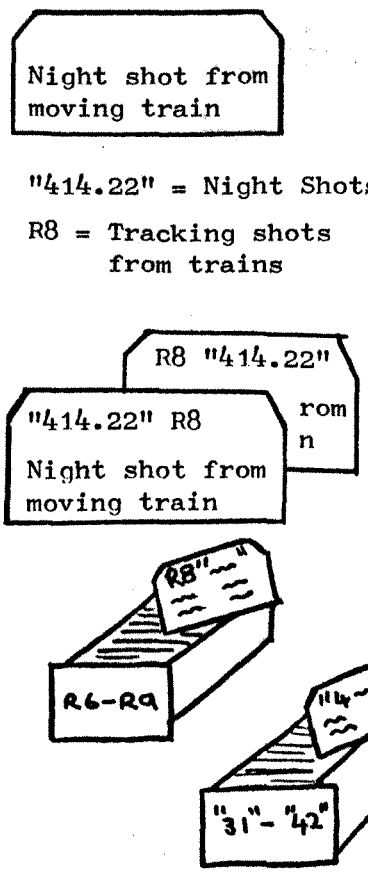
so, strings of the following type can result:-

M65:138.5 VC10 - a VC10 landing

R13(421.2) - aerial shots of London

Practical implementation of the UDC

The generalized UDC storage and retrieval operations are described in figures 1.3 & 1.4, and details of the files central to the maintainance of these processes are contained in figure 1.5:-



Prose card input

↓

Librarian attributes UDC numbers based on card prose

↓

A card is duplicated for each code variant, thus allowing multiple subject access to film

↓

Cards are filed in main UDC catalogue under the attributed codes

figure 1.3 - UDC Storage

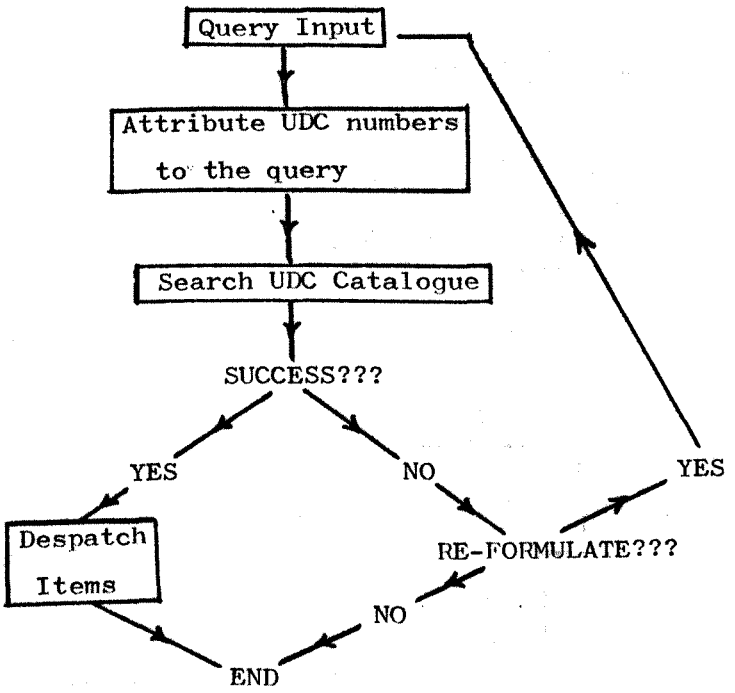


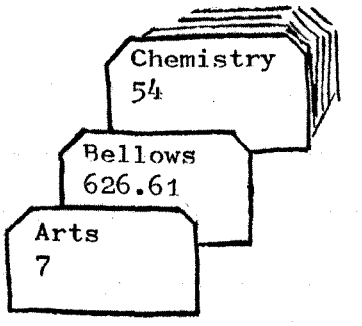
figure 1.4 - UDC Retrieval

SUBJECT INDEX

For translating literal subjects to UDC codes.

Filed: Alphabetically by subject

Used: For attributing UDC numbers to:-
(i) a new card
(ii) a query

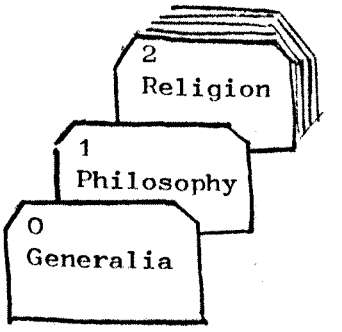


AUTHORITY FILE

For translating UDC codes to literal subjects

Filed: In UDC order

Used: For the maintenance of the UDC structure (but could also be used in retrieval)



UDC FILE

Cards containing literal film item descriptions & the attributed UDC codes

Filed: In UDC order

Used: As the main repository of film descriptions to which any subject search must refer

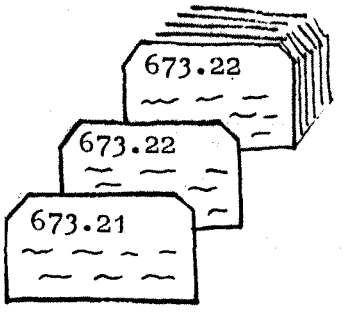


figure 1.5 - The Files

The Subject Index and Authority File are the means by which the codified subject classification is presented in a recognisable (ie. literal) form. It is here that all subjects known to the system, and their associated UDC numbers, are stored. There are, of course, a great many more files, but the three described in fig 1.5 are the only ones central to subject storage and retrieval. An example of a typical entry in the UDC file is given in figure 1.5(a).

34.096:343.77(423.14 Birmingham)	
BIRMINGHAM: BOMBERS IN COURT - ms police motor-bikes (4s)ms policeman by barricade, tilt up policeman on roof of nearby bldng (13s)var s' policemen search cars. Heavy police security for trial of 7 men charged with plotting explosions in the West Midlands between August '73 & August '74.	DATE 10/4/75 DAY NO. POS D 100/75/09 MAG MASTER 100/75/9 CPYRGHT BBC FTGE 22' DURATION 36s BULLETIN 5.45 Nat REP. STOCK ORIG. SOURCE BRMNGHM

figure 1.5(a) - A UDC catalogue card

It is the upkeep of these files that constitutes the maintainance of an effective subject retrieval system, and costs (in 1978) something like £66,000 per year. This does not include retrieval costs, which consist of librarians' wages (around £32,000 per year) and phone bills only. A break down of this sum, together with a few other figures, is contained in fig. 1.6.

figure 1.6 - A Few Statistics

BASIC

Amount of film in vaults = 250 million feet

Rate of growth = 2 million feet per month

LOANS: 300-400 films borrowed and returned each day

CATALOGUES	Size (cards)	Size (drawers)
UDC	400,000	280
TITLE	100,000	70
NAME	100,000	70
SUBJECT INDEX	40,000	36
AUTHORITY FILE	18,000	16

MATERIALS

UDC file grows at 3,000 cards/week, 70 drawers/year.

Between them, the Subject Index and the Authority File grow at 600 cards/week, 15 drawers/year.

Cost of UDC cards = £50/week, & drawers = £1,000/year.

Cost of S.I. & A.F. cards = £7/week, & drawers = £200/year.

Total material cost = £4,000/year

Total SALARY cost of Information Storage = £62,000/year

TOTAL COST of Information Storage = £66,000/year

Filing Time UDC = 15 man hours/week

S.I. & A.F. = 10 man hours/week

FILE MAINTAINANCE 13 man hours/week(intellectual)

(error correction) 8 man hours/week(clerical)

There are three general areas of Film Library operation which have emerged as being likely beneficiaries of computerization:-

(1) File maintenance

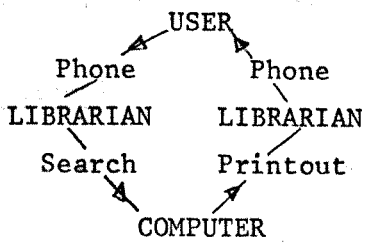
The upkeep of the various files is a mechanical process, both tedious and time consuming. After initial insertion, the UDC file must be periodically checked to allow for disturbances incurred during retrieval searches. Assuming that the filing order can be assimilated by a computer, obvious benefits could be anticipated.

(2) Control of file expansion.

The sheer bulk of the existing files is beginning to pose increasing problems of physical storage. Also, the mobility required of the librarians (especially when more than one file needs to be consulted) should not be ruled out as a possible cause of abandoment of more difficult retrievals. Performing complete storage or retrieval operations whilst seated at a computer terminal would doubtless be more popular with the personnel, and would constitute a major factor in any cost effectiveness discussions aimed at moving towards electronic storage devices. [Microform media could also be considered as a possible solution to size problems alone.].

(3) Query feedback.

I have already referred to the lugubricity of feedback permitted by the present system (see fig 1.2). Given fast retrieval response - which one would certainly expect from a computerized process - the user could be persuaded to maintain the phone link to the Film Library for the duration of the search, and therefore act as the most relevant link in a feedback loop:-



Computerization - things to consider

My comments so far have been of two types; first , those general statements that apply to the systems operated at the Film Library, regardless of the detailed mechanisms upon which they are based; and second , discussion of the specific retrieval method (the UDC) as currently operated.

For the moment, I wish neither to embrace nor reject the UDC as a method amenable to direct computerization, but rather to define system requirements:-

(1) The Information Element

The first input at the storage stage, and the final output as the result of a retrieval search, is the prose description of a film item (currently held on UDC catalogue cards).

(2) Information Storage

The storage system must be such as to attribute some formal description (code, vector or whatever) to the Information Element, so that subject definition is achieved.

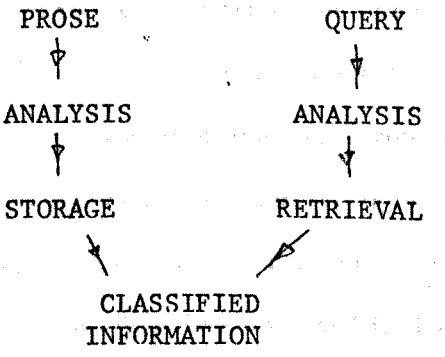
(3) Information Retrieval

The retrieval system must be such as to apply some formal description to the Query, to permit comparison with stored Information Elements, and then retrieve the descriptions most similar to that of the query.

In the practical environment, one can then go on to say that storage by subject class (similar information elements being grouped together) would be desirable, since the opposite would be to imply the necessity of a complete search of the information collection for every query.

Deeper in the pit of practicality, one encounters such truistic demands such as that " the whole system should be as efficient and as error free as possible." At this level also, one appreciates that the system should not be prone to breakdown due to outside circumstances. Already it can be seen that the heights of theory, both abstract and clear, are pre-emptively compromised by the forces of reality. Nevertheless, one can consider the following system view of

the Film Library (or any other retrieval system come to that) without making any assumptions as to whether the various steps are mechanized or manual:-



[Note:- Query Analysis need not follow the same procedure as Prose Analysis]

Starting with a purely manual system, mechanization is most easily directed at the Storage and Retrieval stages, since these consist of conventional Information Retrieval (IR) operations such as searching and sorting, whereas the most difficult step, Analysis, involves the use of intellect or artificial intelligence, and does not therefore emerge as a straightforward task from the computerization standpoint.

In summary, the objectives of computerization can be said to lean in the direction of overall efficiency. A lot can be learned from the existing system, not the least of which is its ability to adequately answer the current range of queries. To computerize the manual system is supportable on two grounds, one no less important than the other:-

(i) In certain areas, manual practice might prove to be superior to mechanized methods in terms of both efficiency and cost-effectiveness.

(ii) One has a duty to avoid replacement of interesting manual jobs by boring mechanical ones.

Of course, doubts surrounding the adequacy of any artificial conceptual framework, such as the UDC, cannot be ignored simply because the services of a computer are harnessed. One can justifiably claim however, that a good deal of the drudgery associated with the upkeep of such a classification could be shouldered by a machine which would, in the process, eliminate those highly disruptive products of human error, such as mis-filing, that constantly, and often latently, reduce system efficiency.

CHAPTER 2

A Statistical Investigation at the Film Library

In order to get a good feel for the retrieval procedures employed at the Film Library, a statistical survey was planned to yield quantitative information on enquiry types, turnaround times and success rates. The survey was centred on the Enquiries section at the Film Library, where all telephone requests for film items are received and dealt with. The amount of information that resulted from the survey was immense, and is contained in a separate report - STAFF[253] (1976). For the purpose of this thesis therefore, I shall be highly selective by way of the results that I disclose.

Before giving detail of the survey however, it is important to acknowledge the TITLE and NAME catalogues, to which reference will be made on a number of occassions. Enquiries can generally be broken down by type (ie. Subject, Title and Name), and whilst the UDC system is used for Subject Retrieval, separate files ordered by:-

(i) Programme Title (TITLE file), and

(ii) The names of well-known personalities appearing in the piece of film (NAME file)

are maintained in order to provide alternative access to stock.

The survey was instigated to achieve three purposes:-

(i) To furnish information on overall aspects of the Film Library, in terms of enquiry types (subject, name or title), busy times, turnaround times & department usages.

(ii) To indicate levels of catalogue usage, efficiency of usage, success rates & actual answering times.

(iii) To give an indication of the way in which the catalogues are used in answering an enquiry and, if it proved possible to do so without introducing distortion, to attempt judgements on the adequacy of the catalogues.

For each enquiry received over the telephone at the Film Library, a form (see fig 2.1) was to be completed to record the salient features of the enquiry and the procedure with which it was handled.

The process of form design was limited by three constraints:-

(i) To yield the required information.

(ii) To permit easy completion by the Enquiry Assistants.

(iii) To permit simple (i.e. card-punch) translation to a machine-readable form, so that analysis could be performed by computer.

It was also required that the forms should carry sufficient information to allow individual enquiries to be followed-up, which explains the presence of the "ENQUIRY & NOTES" and "REQUISITION NUMBERS" sections, which were not read by the computer.

The balance between (i) and (ii) was a fine one, but once achieved, only slight alterations of format were necessary to accommodate (iii).

LEAVE BLANK 1 2 3 4 5 | ENQUIRY RECEIVED :- DATE 6 7 / 8 9 / 10 11 TIME 12 13 . 14 15 A.M. 16 P.M. 17

ENQUIRY ASSISTANT 18 19 | ENQUIRY DEADLINE IMED. 20 ASAP. 21 OTHER 22 | PROJECT NUMBER 23 24 25

ENQUIRER | ADDR. | ROOM | Xtn

SEND TO | LOCATION

ENQUIRY & NOTES

FORM CONSTRAINTS — FILM COL 26 B&W 27 | V.T. 2in 28 CAS 29 | GAUGE 16m 30 35m 31 | * LENGTH 32 * OTHER 33

TYPE OF ENQUIRY — SUBJECT 34 NAME 35 TITLE 36

HOW SPECIFIC? VERY 37 MODERATELY 38 GENERAL 39

WAS A COMPROMISE NECESSARY? 40 *

WAS THE ENQUIRY MODIFIED? 41 *

figure 2.1 - The Enquiry Form

INFORMATION CONSULTED

UDC 42	NAME 43	TITLE 44	PRE-74 V.T. 45	SUBJ. HEAD 46	STRIP NAME 47	STRIP SUBJ. 48	NEWS D.R. 49	PasB etc. 50	* OTHER 51
☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

TIME SPENT IN RESEARCH | 0-5m 52 ☐ | 5-10m 53 ☐ | 10m-1hr 54 ☐ | Over 1hr 55 ☐

SUCCESSFUL?

YES 56 ☐ — WITH WHAT RELEVANCE? — HIGH RELEVANCE 57 ☐ — LOW RELEVANCE 58 ☐

NO 59 ☐ — WHY NOT? — ITEM DOESN'T EXIST 60 ☐ — NOT ENOUGH TIME * 61 ☐ — OBVIOUS SYSTEM DEFICIENCY * 62 ☐

WERE ERRORS ENCOUNTERED?* 63 ☐

REQUISITION NUMBERS :-

ENQUIRY FINISHED :- DATE 64 65 / 66 67 / 68 69 TIME 70 71 . 72 73 A.M. 74 ☐ P.M. 75 ☐

THIS SECTION APPLIES TO U.D.C. AND NAME ENQUIRIES ONLY

LEAVE BLANK					U.D.C. NUMBERS OR NAMES CONSULTED	No. of UDC or Name Hits			No. of Actual Hits			No. of Items Sent		
1	2	3	4	5	10 15 20 25 30 35 40 45 50	52	53	54	55	56	57	58	59	60

A first draft of the form was designed in conjunction with Film Library personnel (including two Enquiry Assistants) and the head of the punch unit at the Open University. In early April 1976, a five day pilot study was run with one Enquiry Assistant, in order to isolate particular difficulties; the important alterations made at this stage are mentioned later.

A final draft was then prepared (again in consultation with Film Library and O.U. staff) and a set of notes prepared to aid the Enquiry Assistants in its completion.

In late April, meetings were held in order to explain the purpose of the forms, and the manner in which they should be completed, to all those members of the Film Library either involved or interested (or both), and for the 24 hours following the meetings, the duty Enquiry Assistants completed the forms as if the survey had begun, so that teething problems might be reduced.

A total of 1,200 forms were produced to support the survey throughout May 1976, of which it was expected some 3-400 would cover subject enquiries. In fact, less than 900 forms were completed during May, and together with those completed over the April 17th trial (which were of sufficient quality to be included in the analysis) the grand total had to be yet further reduced (to 889) owing to the inclusion of several quite useless specimens.

The statistical analysis itself was performed on the Open University's UNIVAC computer, using S.P.S.S. (Statistical Package for the Social Sciences). The raw results of the survey therefore exist in terms of 200 pages of computer print-out, and are available for inspection.

Reference to the form (fig.2.1) reveals that a lot of the information gathered was of a purely routine nature, but in some areas I was aiming slightly under the surface in trying, for example, to trap instances of "enquiry modification". I shall now explain those features of the form that are perhaps not quite so obvious.

For the entries to be made in the ENQUIRIES ASSISTANT section, numbers were allocated to uniquely identify all Film Library personnel likely to answer an enquiry. This identification (which was not used for assessing the efficiency of the Enquiry Assistants) helped in following-up individual enquiries, and monitoring the ease (or lack of it) with which the forms were filled in; this last point requires a little explanation. When vetting the forms prior to their submission for punching, a subjective assessment of the quality with which the forms had been completed was made for every different Enquiry Assistant number. If the analysis then showed that:-

(i) a large number of forms per E.A. = bad completion, and

(ii) a small number of forms per E.A. = good completion

the conclusion would tend to be that completing the forms had been an imposition, and that Enquiry Assistants in the latter category had either had their output reduced as a result, or that they submitted choice examples only. If, on the other hand, no such clear cut inverse relationship between quality and number of forms became evident, it would tend to intimate that no such general imposition existed. In fact, no such relationship did become apparent - the Enquiry Assistant who completed most forms, did so with great proficiency.

FORM CONSTRAINTS may be imposed by the enquirer, which comprise a search parameter that in many cases dominates successful matching on other criteria - eg. if 16mm gauge is demanded, 35mm film will be useless, no matter how perfect the content.

In the majority of cases, this section was left completely blank, and although this does not suggest neglect, it is likely that a "colour-film-16mm gauge" constraint applied even when not recorded as such. This was a fault of the form, in that the norm should not have demanded special action.

In asking the librarians to record the TYPE OF ENQUIRY, it was originally felt that an enquiry could be classed in terms of one, and only one, of three types (subject, title or name). A number of forms however, were ticked as being of more than one type (eg. subject - name) and, on reflection, it was conceded that certain enquiries could not be uniquely attributed to one class. In fact, a fourth enquiry type was encountered, in which the user required special information (technical details for instance) and a box could have been provided to trap this type.

The HOW SPECIFIC? section demanded a subjective assessment, and was therefore liable to the vagaries of the individual's interpretation of what "specificity" means in the context of the UDC catalogue. Specificity is related to the nature of the subject and not the amount of film of a given subject held at the Film Library. For example, film of the planet PLUTO (which is assumed to be rare) would nevertheless indicate a "MODERATE" degree of subject specificity. Title enquiries are usually very specific. It is possible however that a title enquiry is only moderately specific (perhaps even general), if for instance, a particular item is to be chosen from a series of programmes.

The WAS THE ENQUIRY MODIFIED? section arose out of the pilot version of the form, which contained a free-format section headed "Real Enquiry" following immediately after the equivalent "Enquiry & Notes" section. It was hoped that in this way, the enquiry underlying the user's request could be gleaned, since it was felt from the outset that many users constrain themselves by a poor understanding of Film Library operation. A notable example of this occurs when programmes are requested by title because they are thought to contain the item (eg. subject or name item) for which they are looking. In subject areas, they often over - or under- specify, which may be due either to a misunderstanding of Film Library holdings, or poor expression over the telephone.

A particular example of this arose in the pilot study. A request was made for film of "David Cassidy arriving at Heathrow", which could not be satisfied, whilst it transpired that the enquirer simply wanted "film of David Cassidy amidst a mob of fans", and which could be satisfied . The enquiry was therefore modified to achieve success without compromising the user.

The somewhat idealistic belief that a "real enquiry" could be solicited was frustrated by the speed with which Enquiry Assistants often have to work, and so this was replaced by the "tick and give details" section to be seen in the final version. This section was seldom completed, but the few Enquiry Assistants who did use it, did so with a good understanding of its importance.

When the survey was begun, the initial reaction of the Enquiry Assistants, though not hostile (not quite), was adequate indication that a somewhat shorter form would have been welcome, and if a continuation of the survey were proposed, the amount of information solicited from the Enquiry Assistants could be trimmed to trap only the more vital details of an enquiry. Selected results of the statistical survey appear in figures 2.2 to 2.19, and each is followed by a short discussion of its importance.

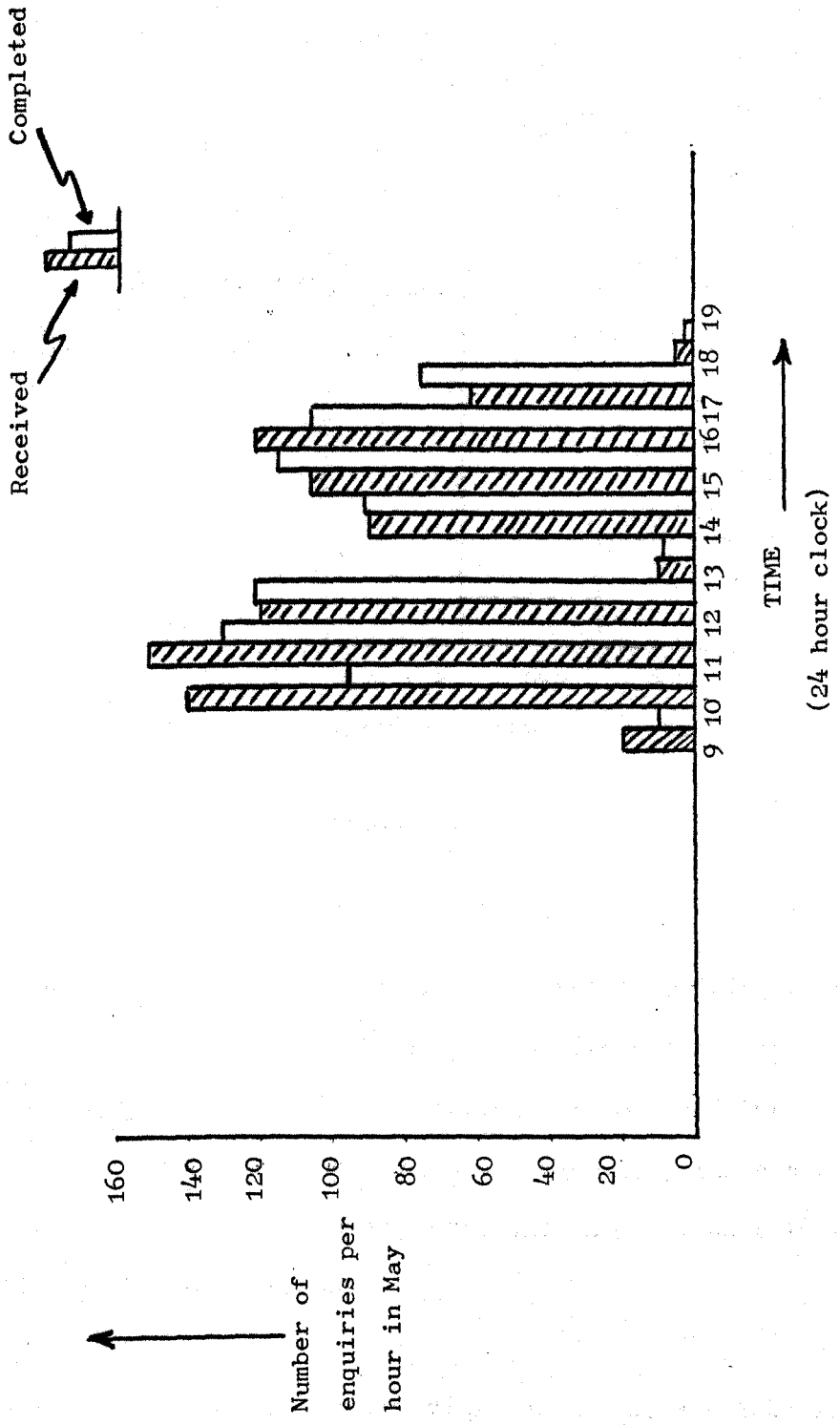


figure 2.2

Distribution of enquiry load throughout the day, totalled for the surveyed month of May 1976. Even allowing for missing cases, it would appear that over the 21 working days (Mon. - Fri.) that the survey covered, not more than 10 enquiries were received in any one hour.

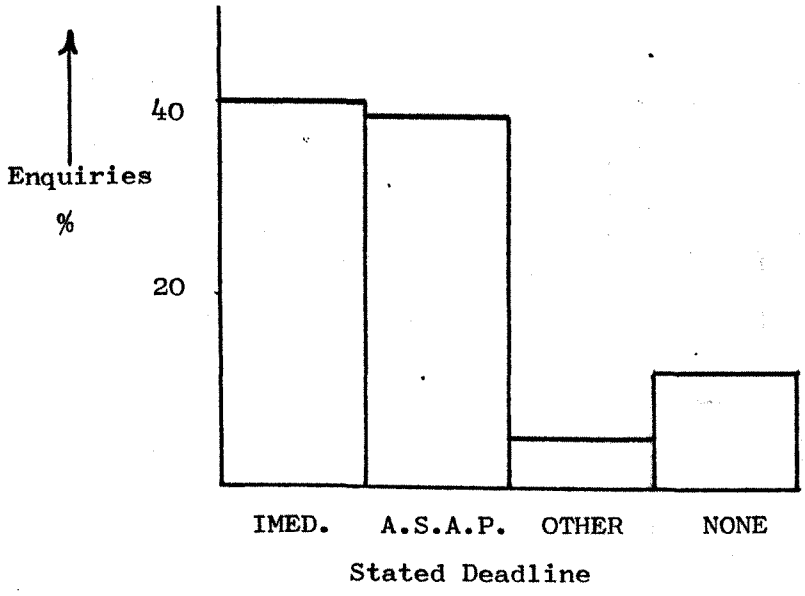


figure 2.3

Distribution of stated Deadlines. It is not surprising that most users feel disposed to state the urgency with which they require their particular enquiry to be treated, but the need for some degree of haste in almost 80% of the incoming enquiries, imposes a constraint (by way of user's expectation) on the manner in which the Film Library has to operate.

Some 40% of enquiries had an immediate deadline, and this sort of level (30-50%) was maintained throughout the day (9am to 5pm) with the exception of the hour between 1 and 2 p.m., when 76.9% of enquiries had an immediate deadline. As only 13 enquiries were received at this time throughout the survey, this figure need not be of any great relevance. Of the 13 enquiries received between 6 p.m. and 9 a.m., nine had immediate deadlines.

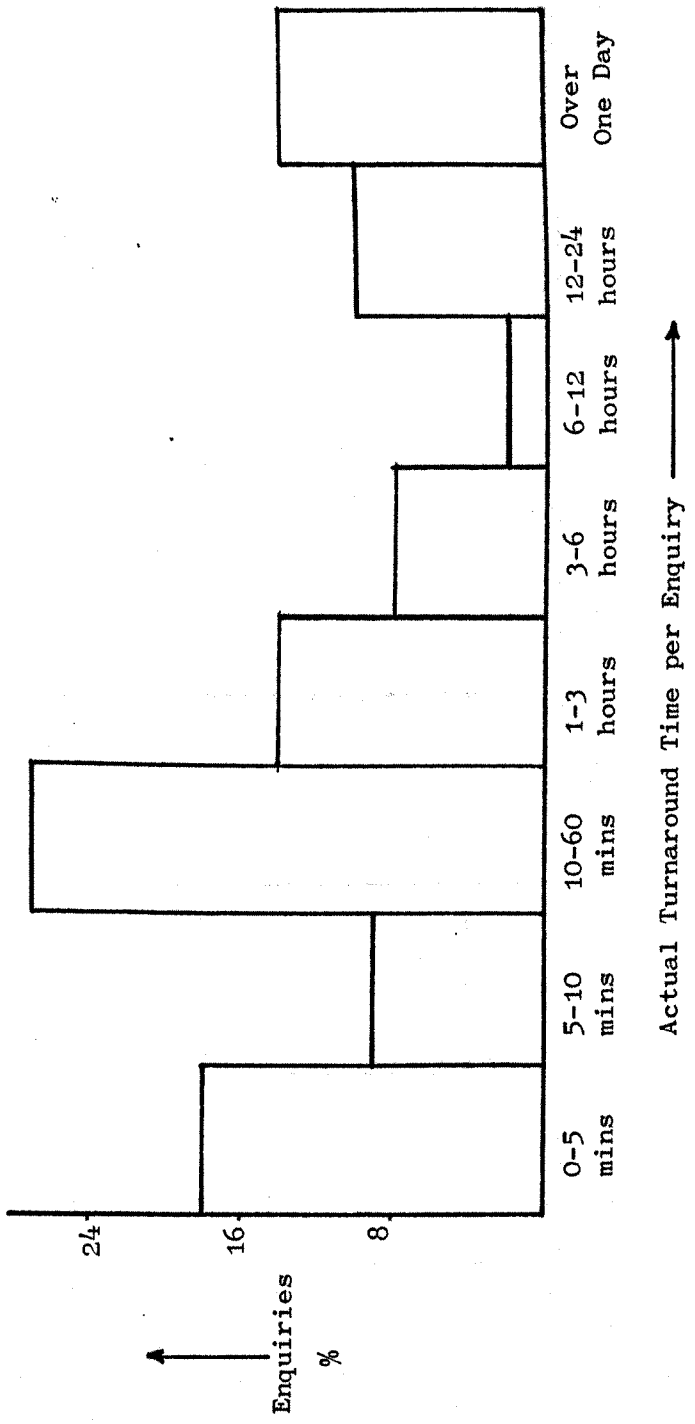


figure 2.4

Distribution of Turnaround Time per enquiry, computed as the difference between time of receiving and time of finishing an enquiry. 53.5% of enquiries were dealt with in under one hour, whereas 75% achieved "same day" turnaround (under 12 hours).

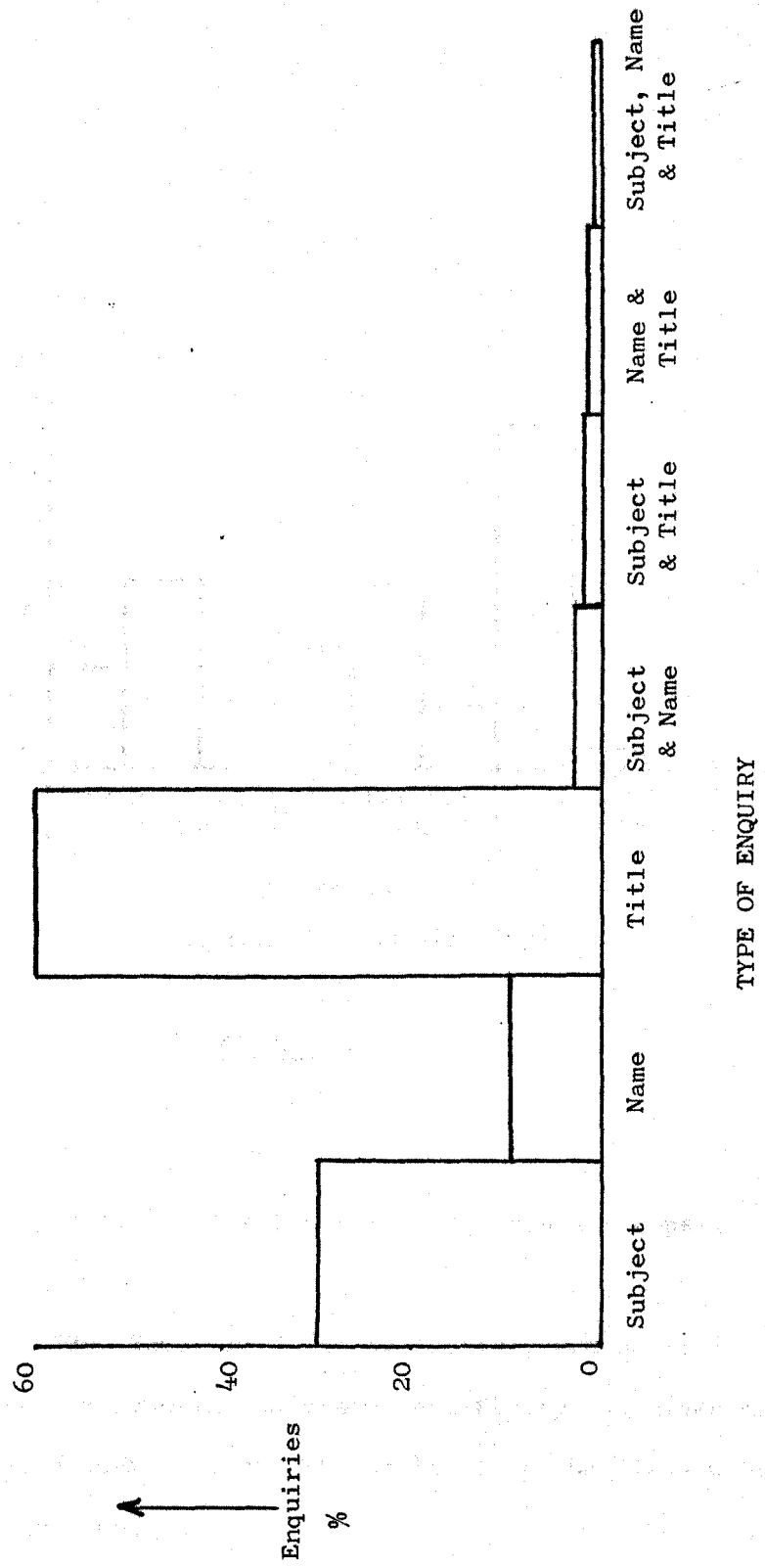


figure 2.5

Distribution of Enquiry Types. As expected, title enquiries predominated, and the predicted "one third" subject enquiries was almost exactly realised. 5.3% of the total were multiple-type enquiries (a useful improvisation by the Enquiry Assistants).

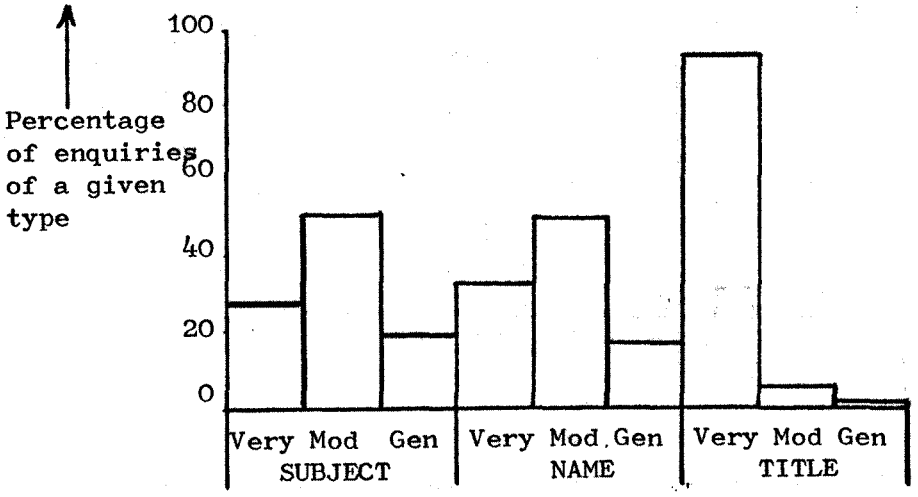


figure 2.6

Distribution of Specificity for the three main subject types.

The assessment of "How Specific?" was subjective, and so it is possible that the box denoting moderate specificity was given more attention than it deserved, since it was likely to be "less wrong" than one of the extremes.

Twelve enquiries were modified (1.3%) of which ten were SUBJECT type and five were very specific. Over-specificity was expected to be a prime cause of enquiry modification.

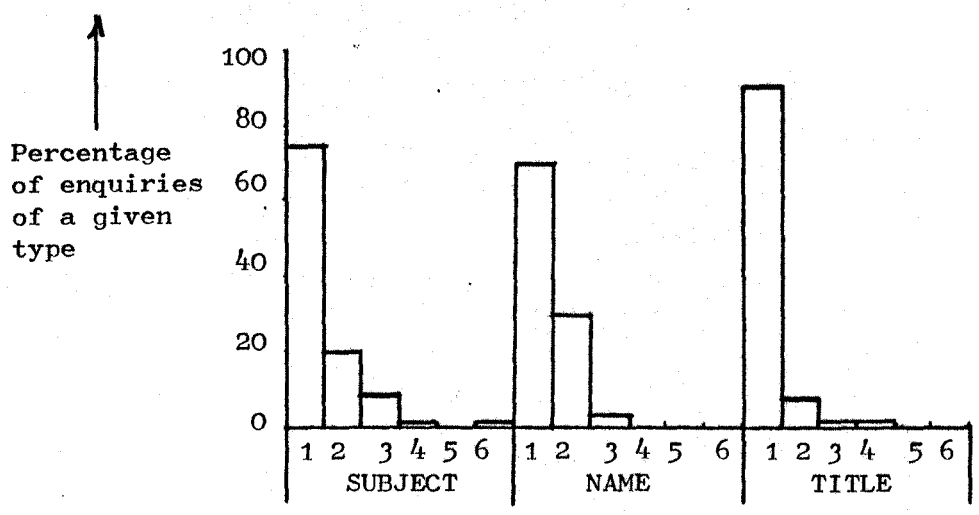


figure 2.7

Distribution of Number of Files Consulted for the three main enquiry types. Details of the usage of these files are given in Figs. 2.11 to 2.13.

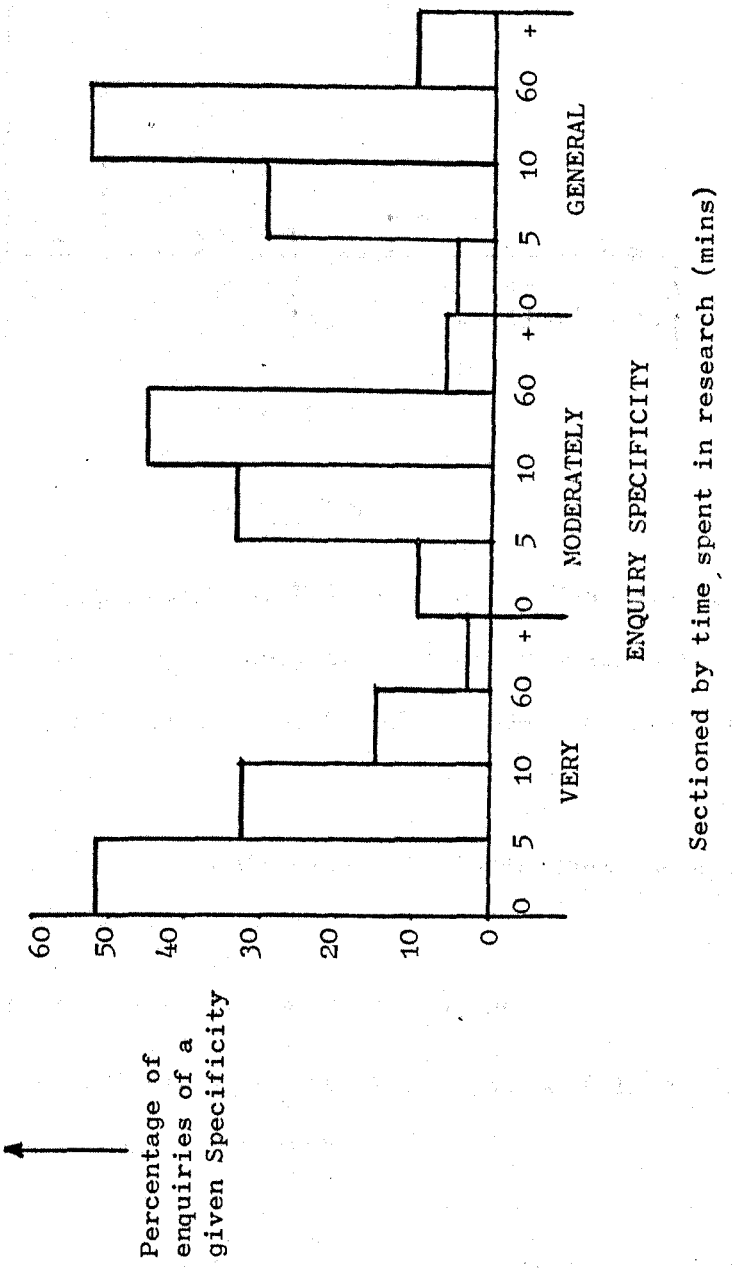


figure 2.8

Distribution showing the effect of Specificity on Time Spent in Research. The expected trend is obvious, namely that decreased specificity leads to increased research time.

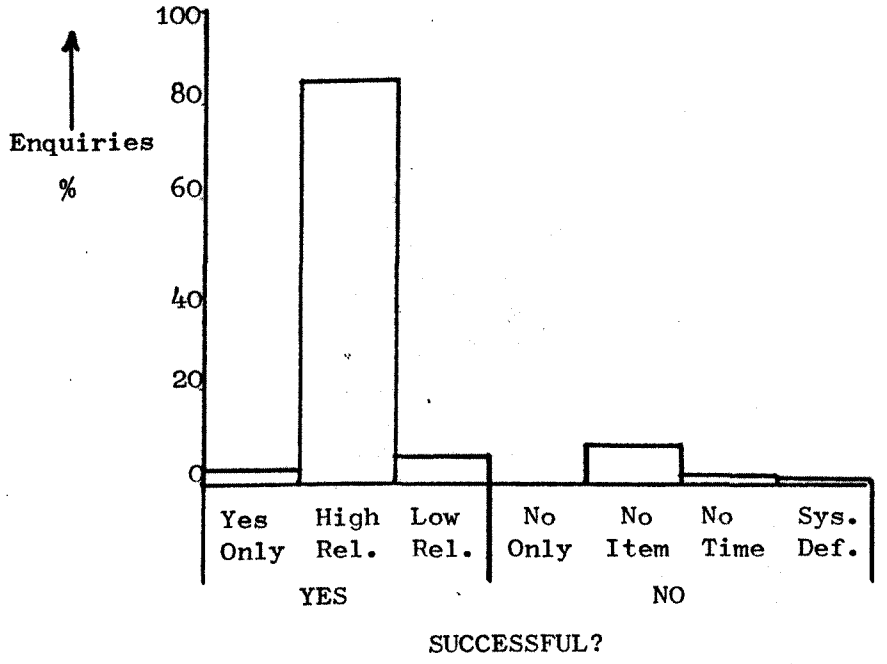


figure 2.9

Distribution of overall "SUCCESSFUL?" ratings. "Yes Only" and "No Only" refer to forms on which only the Yes or No boxes were ticked without further qualification. 91.9% of the 889 enquiries were answered successfully.

[Note! - The 7.1% "Item doesn't exist" failures could have arisen in one of two ways:-

(i) latent retrieval system defficiency, or

(ii) genuine non-existence of item eg. "Send me film of the Titanic arriving in New York".

This last example (although it actually happened) is not typical. For the most part, genuine non-existence of film cannot be detected, and retrieval system failure is always a possibility.]

The effect of several variables on success was computed, though not felt to be particularly worthy of plotting. For instance, enquiry deadline had no effect upon success, with not even "immediate" enquiries prompting a "no time" failure; nor could the number of form constraints be seen to have any destructive effect upon success.

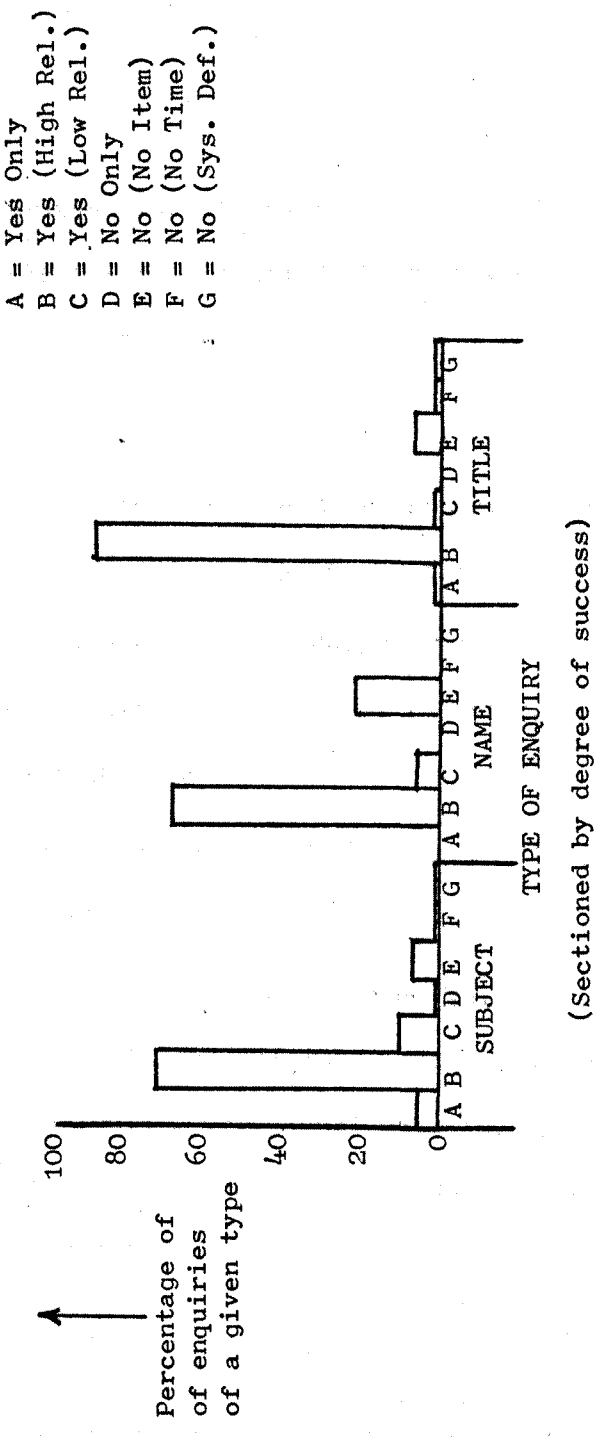


figure 2.10

Distributions of Success ratings for the three principal Enquiry Types. 90.9% of subject enquiries were successfully answered (7.3% "no item" failures) and 93.5% of the title enquiries (5.8% "no item"). 76.1% of the name enquiries were successful.

Catalogue Usage

As will be noted from the reverse side of the form (fig. 2.1) a number of files were available for consultation by the Enquiry Assistants, to which I have not hitherto alluded:

SUBJECT HEADINGS: An alphabetically ordered subject file that preceded the UDC system at the Film Library.

STRIP SUBJECT: An alphabetically ordered subject file (hung on a wall!) that preceded the Subject Headings file.

STRIP NAME: A wall mounted file that preceded the present NAME file.

P as B: Sets of notes detailing the "Programme as Broadcast", ordered by transmission date. These contain televisual details, programme timings etc., that are recorded at the time of transmission.

The fact that a given catalogue was used in answering an enquiry does not mean that the same catalogue led to its success, but rather that the said catalogue contributed to the success of the enquiry, since another catalogue used in the search may have been entirely responsible.

Figure 2.11 is a table in which Catalogue Usage is correlated with Enquiry Type. Of the three main catalogues (UDC, Name and Title) it was the Name catalogue that enjoyed the greatest diversity of use, 16.9% of the references to it being brought about by what were originally subject enquiries.

Figure 2.12 is a table correlating Catalogue Usage with Success. During the whole course of the survey, the UDC catalogue was used in 257 enquiries, the Name catalogue in 84, and the Title catalogue in 517. The figures for the remainder were: Pre-74 V.T. 4, Subject Headings 20, Strip Name 12, Strip Subject 15, News Day Record 10 and P as B 5.

The "Other" category, which covers phone calls, memory etc., proved itself both quick and useful, in that it was invoked in 48 cases, most of which would have been last resorts.

Figure 2.13 gives details of simultaneous use of catalogues, and is of interest when considering computerization. Certain joint usage is to be expected - e.g. Strip Subject with UDC (93.3%), Subject Headings with UDC (85.0%) etc. - whereas certain other conjunctions are less obvious, but not surprising - e.g. P as B with Title.

The large use of other files when the Name catalogue was consulted (36.9% UDC and 32.1% Title) can - if the 84 name enquiries are considered an adequate sample - be blamed partly upon its associated success rating, which was lower than for both UDC and Title catalogues.

Probably the most striking feature to come out of this table is the set of high values contained in the Title column, since this indicates the hallowed position held by the Title catalogue in providing a source of basic reference.

figure 2.11

Figures are percentages, totalling vertically to 100% in each column

ENQUIRY TYPES \ CATALOGUES	UDC	NAME	TITLE	Pre-74 VT	Subj Head	Strip Name	Strip Subj	News D.R.	PasB etc.	Other
SUBJECT	84.0	16.9	7.5	0	75.0	8.3	66.7	50.0	20.0	17.0
NAME	1.2	50.6	1.2	0	0	50.0	13.3	0	0	2.1
TITLE	3.5	4.8	86.1	50.0	5.0	0	6.7	30.0	80.0	72.3
SUBJ/NAM	5.4	18.1	1.0	25.0	15.0	33.3	13.3	10.0	0	2.1
SUBJ/TIT	4.7	1.2	2.5	0	0	0	0	10.0	0	2.1
NAME/TIT	0	4.8	1.2	0	0	8.3	0	0	0	4.3
S/N/T	1.2	3.6	0.6	25.0	25.0	0	0	0	0	0

figure 2.12

Figures are percentages, totalling vertically to 100% in each column

SUCCESSFUL ???	CATALOGUES										
	UDC	NAME	TITLE	Pre-74 VT	Subj Head	Strip Name	Strip Subj	News D.R.	PasB etc.	Other
Yes Only	5.3	7.2	1.2	0	5.9	8.3	8.3	0	0	0
Yes High Rel	73.6	66.3	90.4	50.0	41.2	41.7	58.3	100	100	66.7
Yes Low Rel	11.4	8.4	2.4	0	35.3	33.3	25.0	0	0	2.2
No Only	0.4	0	0	0	0	0	0	0	0	0
No No Item	8.9	18.1	5.9	50.0	17.6	16.7	8.3	0	0	26.7
No No Time	0.4	0	0.2	0	0	0	0	0	0	2.2
No Sys.Def.	0	0	0.4	0	0	0	0	0	0	2.2

figure 2.13

Figures denote the percentage of enquiries using a catalogue (1) that also use a catalogue (2)

eg:- 12.1% of enquiries using the UDC catalogue also use the NAME catalogue.

CATALOGUES (1)	CATALOGUES (2)									
	UDC	NAME	TITLE	Pre-74 VT	Subj Head	Strip Name	Strip Subj	News D.R.	PasB etc.	Other
UDC	100	12.1	22.6	1.2	6.6	1.9	5.4	2.3	0	2.7
NAME	36.9	100	32.1	2.4	6.0	14.3	4.8	1.2	0	6.0
TITLE	11.2	5.2	100	0.6	1.4	0.6	0.6	0.4	0.6	0.6
Pre-74 VT	75.0	50.0	75.0	100	25.0	0	0	0	0	25.0
Subj Head	85.0	25.0	35.0	5.0	100	20.0	50.0	5.0	0	5.0
Strip Name	41.7	100	25.0	0	33.3	100	33.3	0	0	16.7
Strip Subj	93.3	26.7	20.0	0	66.7	26.7	100	13.3	0	6.7
News D.R.	60.0	10.0	20.0	0	10.0	0	20.0	100	0	0
P as B etc.	0	0	60.0	0	0	0	0	0	100	0
Other	14.6	10.4	64.6	2.1	2.1	4.2	2.1	0	0	100

"String Searches" are those using NAMES and UDC codes in their respective files. The remaining figures exclusively concern such searches. 503 string searches (Name, and UDC) were associated with the 889 enquiries; in other words, 503 entries were made in the "UDC and Name" section at the end of the form. For the purpose of this analysis, UDC strings were further divided into:-

(i) normal strings (denoted "0-9 strings"), and

(ii) strings beginning with one of the motion designators (R and M).

These "motion designators" were introduced at the Film Library some time ago to enable description (in UDC terms) of both subject motion (M) eg. "Take-off" and "landing", and shot type (R) eg. "air-to-ground", "night shots" etc.

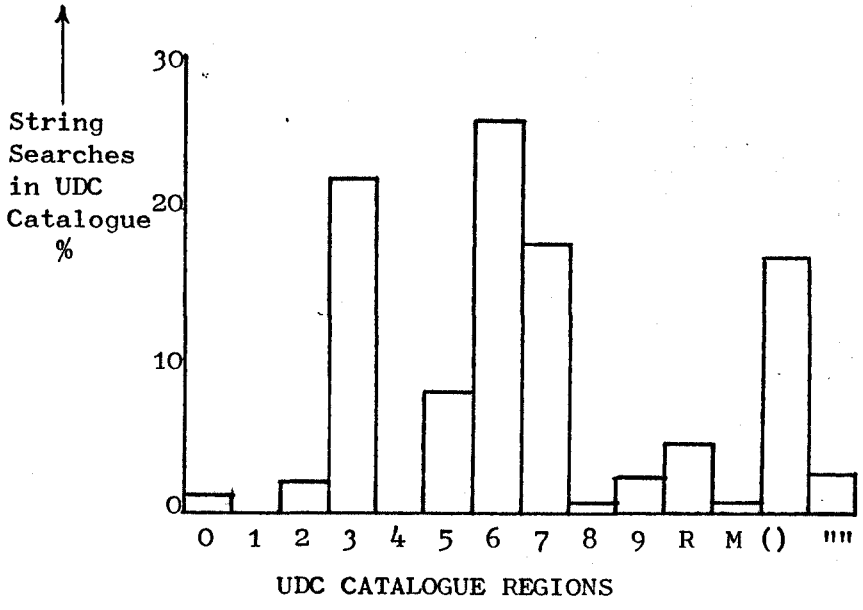
In what follows, the term RELEVANCE is used. In standard Information Retrieval parlance, this quantity is defined thus:-

$$\text{RELEVANCE} = \frac{\text{Number of Relevant Items Retrieved}}{\text{Total Number of Items Retrieved}} \times 100$$

High Relevance is, therefore, desirable. In the survey, Relevance was computed as follows:-

$$\text{Relevance} = \frac{\text{No. of Relevant Items Retrieved}}{\text{No. of items retrieved by UDC search}} \times 100$$

which is identical to the standard definition but for browsing. After a first search in the UDC catalogue, it is standard practice for the librarians to browse in the target area in order to refine the search and, in so doing, Relevance may be boosted to above 100%.



Main UDC classes:- 0 = Generalia
1 = Philosophy
2 = Religion
3 = Social Sciences
4 = Languages
5 = Natural Science
6 = Applied Science
7 = Arts, Recreation and Sport
8 = Literature
9 = Geography, History and Biography
() = Location
"" = Time

R & M denote motion

figure 2.14

Distribution of Strings throughout the main UDC regions. As well as the main class (0-9) and motion designators (R & M), this histogram also sorts and strings beginning with brackets (denoting race or geographical location) and quotation marks (denoting date to which footage refers). The prime areas of usage were 6(26%), 3(22%), 7(18%) and Brackets (17%).

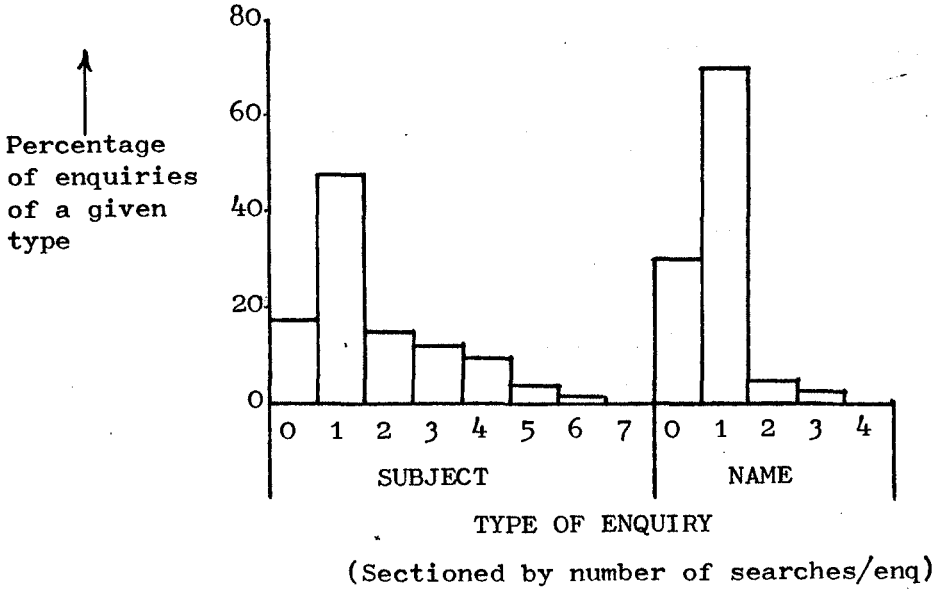


figure 2.15

Distribution showing the number of string searches per enquiry, for subject and name types only. 549 of the 889 enquiries required no string searching whatsoever, and so the 503 searches were spread over 340 enquiries (averaging 1.48 searches per enquiry). Of these 340, 167 necessitated 1 search, 47 - 2 searches, 32-3 searches, 19-4 searches, 6-5 searches, 3-6 searches and 1-8 searches. In this histogram, the enquiries requiring zero searches were those mostly solved by reference to Subject Heading and Strip Subject catalogues for subject enquiries, and to the Strip Name catalogue for name enquiries. The fact that the overwhelming majority of name enquiries were treated in one search is to be expected - names leaving less room for interpretation than concepts: there was no recorded evidence on the forms of searching for a mis-spelt name, though this probably happens quite a lot, under which circumstances a shift to the right (away from the one-search-name-enquiry) would result. The largish zero-column for name type should not lead to any rash conclusions, but lack of confidence (in cataloguing) is a possible explanation.

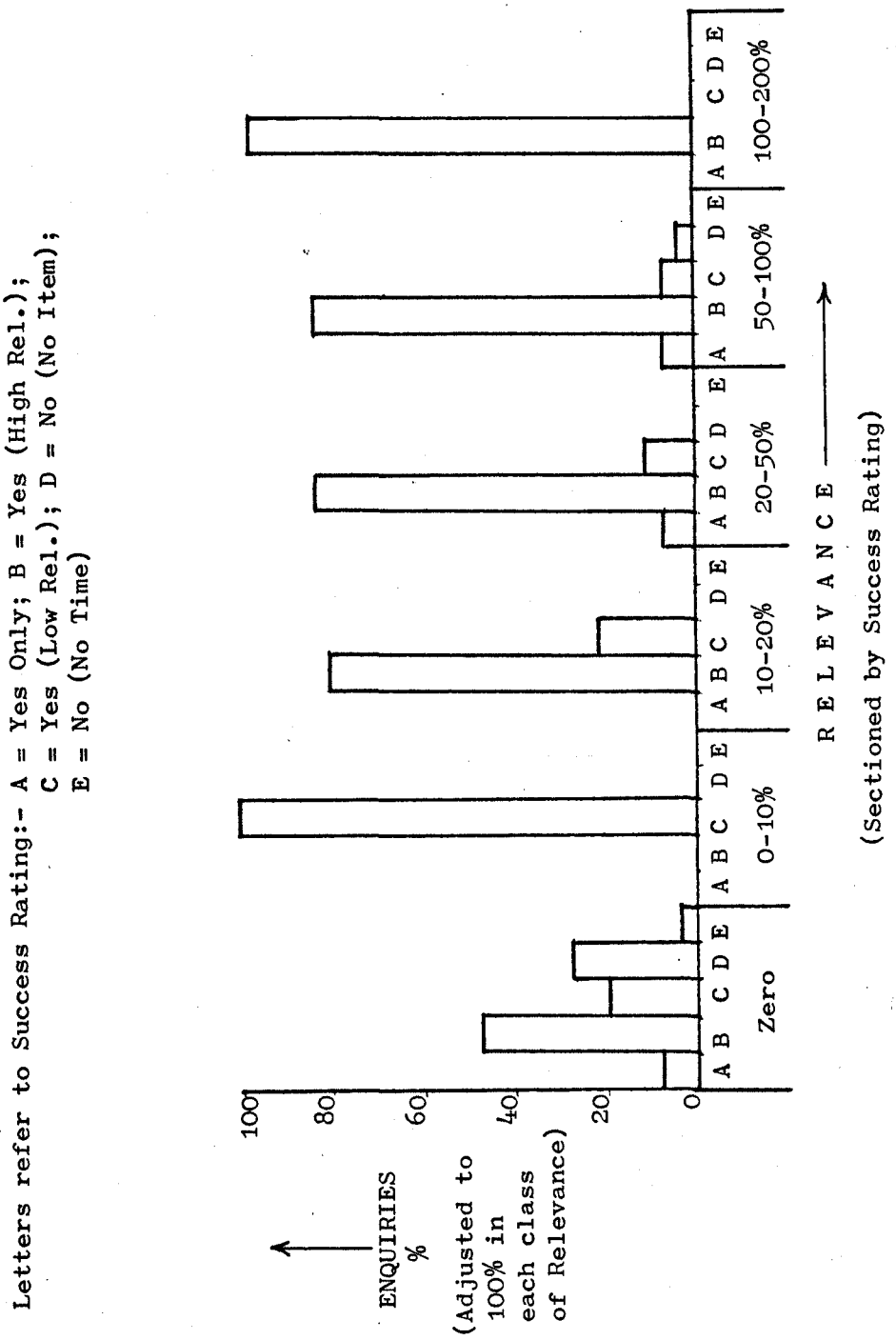


figure 2.16

Distribution showing the relationship between Relevance and Success, for UDC (0-9) string searches only. As one would hope, success improves with increasing Relevance. The note at the beginning of the "Catalogue Usage" section also applies here. A given search must be considered contributory - to the overall efficiency of an enquiry, and not solely responsible, since it may have been only one of several searches employed in achieving the result .

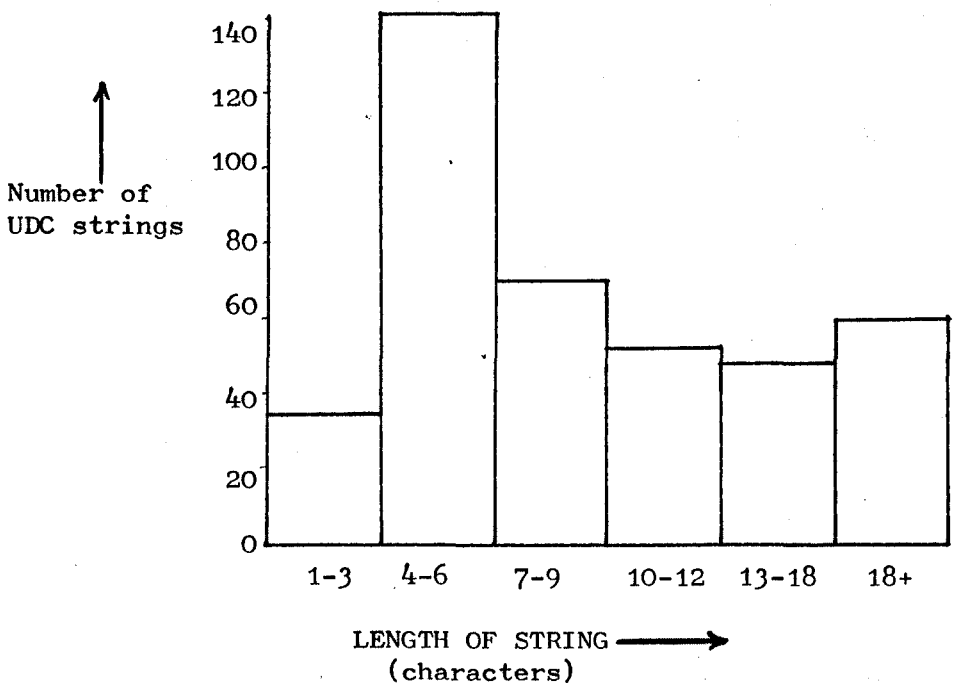


figure 2.17

Distribution of the length (in characters) of all the UDC strings employed in searches. The majority of such strings - some 35% - consisted of between 4 and 6 characters, which could be taken to indicate a preference for general UDC searching, followed by browsing refinement.

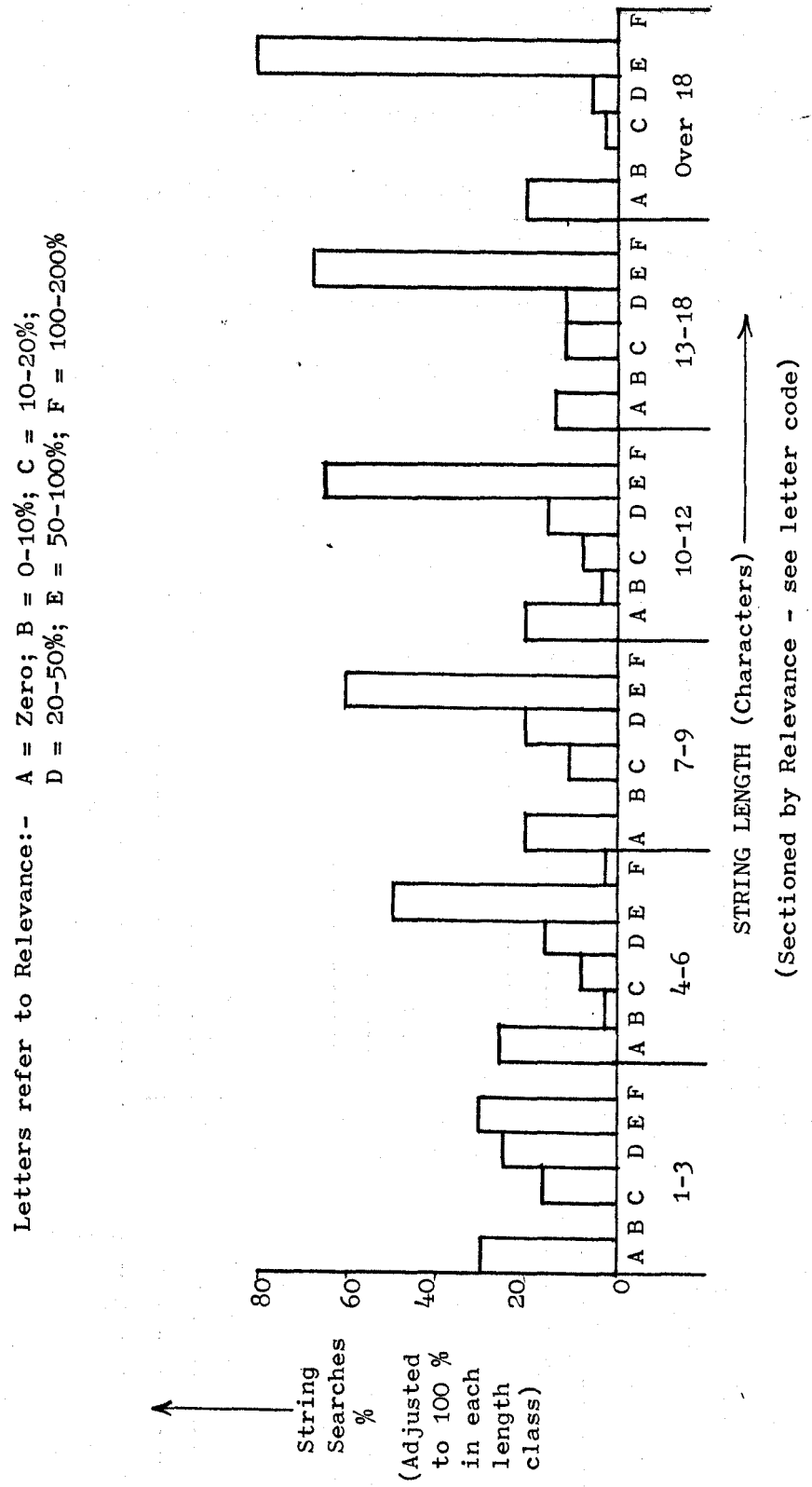


figure 2.18

Distribution showing the relationship between String Length and Relevance for UDC (0-9) string searches only. The trend here is very noticeable - increasing string length (string specificity) leading to a marked improvement in Relevance (and hence, to a large degree, success - see Fig.2.16). The fear that increased string length

might reduce Relevance by imposing over-strict criteria (the longer the string, the harder to match) would appear to be groundless, but this is probably in no small part due to the Subject Index (from whence search strings are taken) which will predominantly contain strings that are likely to succeed. Of course, string length can only be taken as a rough indication of subject specificity.

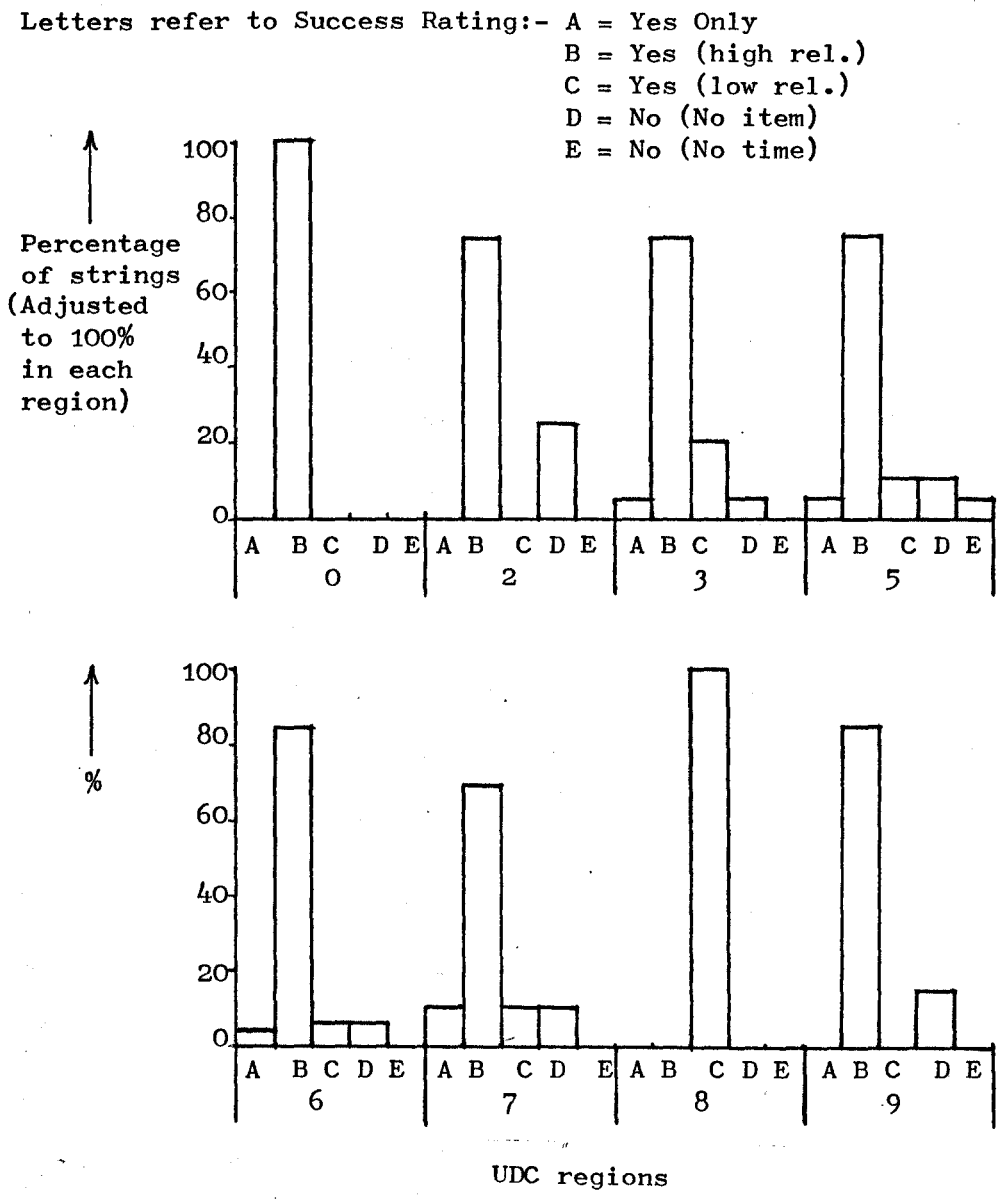


figure 2.19 (cont'd on the next page)

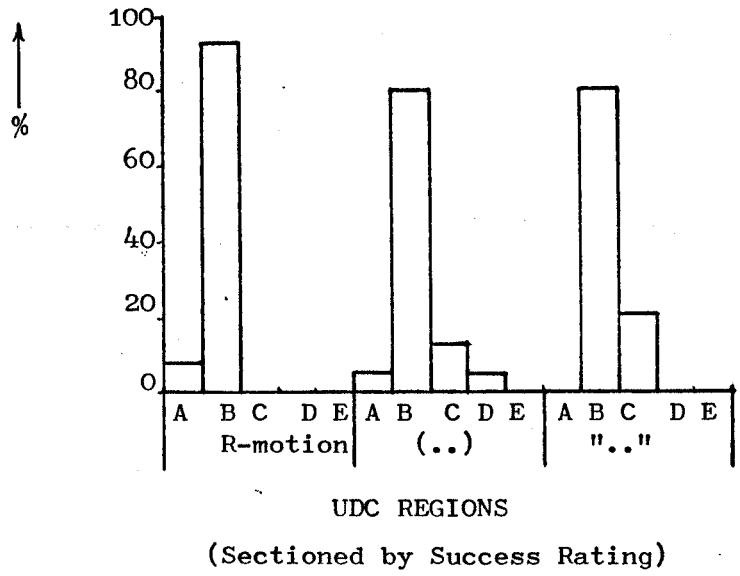


figure 2.19 (cont'd)

Distribution of Success, sectioned by UDC region (including Motion). Of the well populated areas, 7 (composed of 68 good readings) has the lowest high relevance success at 70.6% (although 10.3% are Yes Only), but even this commendable value is dwarfed by the 85.6% value for area 6 (having 90 good readings).

Discussion of the Statistical Survey

Some of the results gleaned from the results will doubtless prove useful in certain technical aspects of retrieval system design, and other results will be mentioned later in this thesis.

The central importance of the TITLE catalogue was appreciated before the survey was begun, but the high efficiency of the UDC catalogue was illuminating, especially since this catalogue provides one of the largest stocks of data for film subject research in the country.

On a more practical note, the Table (Fig.2.13) which presents details of the simultaneous use of catalogues - together with the

other diagrams in the section on catalogue usage - provides an vital indication as to the information stock upon which enquiries draw - an important factor.

As a note to the librarians currently using the UDC system, I might stress the high success rates associated with the more specific searches (Fig. 2.18), especially since a tendency to the opposite obviously exists (Fig. 2.17).

Use of the NAME file often emerged as being less likely to produce success than either the UDC or TITLE files, but that's not to say that it doesn't provide a vital means of access to stock.

It is to be regretted that "Enquiry Modification" was not identified in as many cases as the librarians know the phenomenon underlying it to occur. The user's tendency to merge what he knows he wants with where he thinks it can be found is one that reflects on human nature rather than on Film Library operation, and the result of this is manifest in the high use of the Title catalogue. Who knows the extent of the retrieval process shouldered in the production departments, with users trying to relate shots they remember to programmes they've forgotten? No doubt such enquiries could have been isolated given a longer questionnaire and enthusiastic interrogation of the users, but it is hard to say which of these moves would have been the more unpopular, and hence the more efficacious in ruining the survey.

In fact, assessment of a retrieval system (which is what the survey attempted) is hard enough in the tests carried out of late in manageable computer-based collections (I have restrained myself from using the word "contrived" in this context) and, as long as the user of a retrieval system has preconceptions about that system, then he should be viewed as an integral part of it. The ideal of course, is to test

the system without the user, a strategy which would yield perfect, and, at the same time, perfectly meaningless results. To lead the user out of a blind alley is, on the other hand, amongst the most creditable of retrieval system qualities, and is one which the Film Library Enquiry Assistants are in a position to achieve; the aim of the survey, however, was not to assess the Enquiry Assistants.

In closing this section on a more practical note, I can point out two indications given by the survey that reflect directly on the present manual system:

(1) The most commonly accessed areas of the UDC catalogue were 3 (Social Science), 6 (Applied Science) & 7 (Arts, Recreation & Sport). Assuming the survey to be representative, one can therefore immediately make the inference that areas covering "concrete" subjects are those to which reference is most frequent. Whether or not this means that one can expect to see little philosophy, theosophy etc. on BBC TV is neither here nor there; that an enquiry is more likely to concern film of Concorde than film of a discussion on the Marxian dialectic is again of little consequence to my researches. The important point is the intuitive one, and it is that most people are happier in dealing with and expressing the concrete than the abstract, since request for film of a "thing" will result in a simple YES/NO answer most of the time, whereas a request for film of a less tangible commodity is:

(a) harder to phrase,

(b) harder to communicate and

(c) less likely to yield a straightforward answer.

It has been suggested that this fear of the abstract could lead to poorer indexing and UDC schedule maintenance in abstract regions, the natural tendency being to go for generality rather than committing oneself to a mistake. It is to be regretted that attempts to achieve high specificity are more likely to produce errors in abstract areas, due to muddling of concepts; this inevitably leads to non-committal generality and hence inundating recall (low relevance).

The precise definition in concrete areas (a plane is a Boeing 747 or it isn't) means that one can achieve high indexing specificity in such regions without incurring errors. One would therefore expect to experience greater retrieval success (higher relevance) in concrete areas.

In some experiments by SABEL[21] (1962), catalogue usage has been related to indexing effort in the expectation of finding some correlation by subject area, based on the supposition that well-fingered subjects will be served by a greater intensity of classification effort. No statistics were available to permit such a comparison to be made at the Film Library, and it was heartily denied that any such correlation would be desirable - storage being independent of expectation of retrieval.

(2) The low dependence and relatively low success ratings attached to the Name catalogue have already been mentioned, and a connection between these two characteristics would seem obvious - an inefficient source of reference would hardly attract excessive use. The YES/NO nature of name searches is the probable cause of this

inefficiency, since no recourse to browsing would be fruitful once various spellings and aliases had been investigated - Mr. A. Jones need bear no relation to Mr. B. Jones. If, however, names were to be incorporated into the UDC catalogue itself, with the actual name subordinate (i.e. following) the more general classification of the person, then browsing would be constructive and, what is more, greater complexity could be attached to the name in terms of action, location etc. e.g.:

XXX.XX WILSON, H YYY.Y

where XXX.XX is the UDC no. for a politician
and YYY.YY is the UDC no. for smoking a pipe.

Without wishing to deride the results of an investigation that involved rather a lot of work, it must be said that "interesting" is perhaps a better word than "important" in describing the outcome. Histograms such as that charting variation of enquiry load during the course of a day are nice to see, and might prove to be of real importance in making estimates of machine usage in the event of computerization. In general however, attempts to make more profound judgements of the retrieval system itself, were thwarted by the ability of the system to deflect any probing that went beyond the input/output interface. Without being personally acquainted with the whole of the Film Library stock (which, if it were possible, would render any formal retrieval system unnecessary). I was unable to penetrate that inner sanctum of secrecy that a retrieval system, once used exclusively to define the information

domain, can dominate so completely. The optimistically postulated "following up" of enquiries - which I hoped might produce intimations of defective storage and retrieval strategies - proved singularly ineffectual in the face of the vast amount of data that confronts any search, other than that to which the system has been tailored (UDC, Name or Title). My expected panacea - the use of files not immediately applicable to the original request as stated - was invariably anticipated by the librarians, and so my attempts to find ways round retrieval failures were themselves failures. I refrain from acclaiming the success ratings too highly, since although apparently encouraging, I have found no reason to dismiss my doubts that a retrieval system protects itself from the absolute manifestation of such quantities, other than when measured in small manageable subsets where the ideal result is known before the question is asked.

CHAPTER 3

A Literature Survey

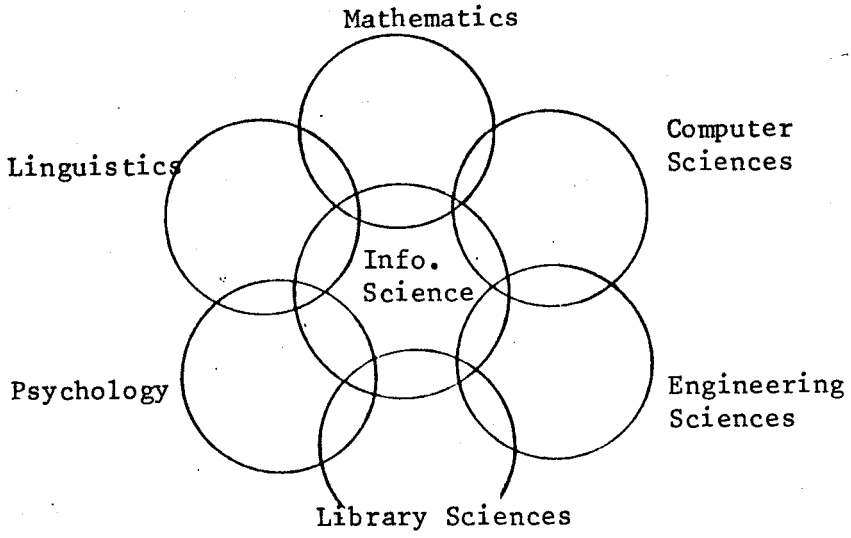
This brief review concerns library automation in general, but with a particular emphasis on UDC-based systems.

In writing this Chapter, I have seldom made any distinction between document retrieval systems, and retrieval of information stored on other media - notably film. There are two reasons for this :-

(i) At the BBC Film Library, a description of the film is recorded exclusively upon the catalogue cards and, therefore, it is reduced in practice (neglecting those rare occasions on which film is viewed before despatch for reasons other than quality assessment) to a document retrieval system in which the catalogue cards themselves, abstracts as it were, become the documents; and

(ii) Literature concerning film library automation is rare - a paper by COX[1] (1961) being a rare exception.

In the course of reading the papers upon which this survey is based, I encountered the description of information science as a "soft science" (O'CONNOR[2] - 1973), that is, one lacking the rigorous identity that only time can bring, time and the gradual process of definition brought about as much by work at the periphery of a subject as at the centre. An idea of the multi-discipline of information science is given in figure 3.1 which is due to OTTEN[3] (1970). Although the detail of this description is unimportant, the need to recognise the present fluidity of the subject is paramount, & it is for this reason that the terminology needs a little clarification.



Information Science and Related Sciences (Schematic two dimensional representation of an n-dimensional relationship--only the most important related sciences are shown)

figure 3.1 - OTTEN[3] (1970)

A hundred years ago "library science" was "information science", man's concept of information being completely linked to the stores of recorded knowledge. The onset of communication theory however, has impinged upon information science both terminologically and conceptually, with the result that the hitherto unambiguous term "information" now differs greatly in interpretation (though hopefully not in basic meaning) when used, on the one hand, by a librarian, and, on the other, by a mathematician (eg. SHANNON[4], whose mathematical measure of information dates from 1949).

A similar ambivalance applies to information retrieval. In this case, the blanket definition is not in dispute, but the librarian's idea of retrieving information, which may imply assigning and searching on a Dewey code for example, is not shared by a computer scientist, who sees it in terms of database design, tree structures and the like.

Having studiously built up the framework of a potential confusion, I now intend to sweep it aside without compunction, by saying that in practice no such confusion arises. When reading a paper in the Journal

of Documentation say, reference to information science (in the library sense in this case) is clearly recognised, and one rapidly appreciates that understanding of "binary digits" and "discrete noiseless channels" can be ignored in discussing the application of UDC auxilliories for example. In other words, I shall write freely about "mechanized IR systems" (which would be tautology to a computer scientist) in one breath, and "hash code retrieval" in the next. In fact, the merging of the two aspects of the science is occurring so rapidly (in definition if not in rigour) as to make this argument almost unnecessary.

As long as animals have possessed memories, they have practiced information retrieval. The recording of knowledge - from cave paintings and hieroglyphics to early writings - caused little impact on the way information was organized for retrieval, although the mode of access (and possibly the accuracy) was revolutionized by dependence upon external recording media. Even when collections where first established, it is unlikely that the form of knowledge was considered as distinct from the knowledge itself, and only when large collections were begun is it probable that the method of organization became important. Perhaps at this stage, the term "library" can be introduced to describe an information collection based on a rigid (meaning reliable) structure or catalogue, however primitive. Somewhere between Aristotle and the mid - 19th century, we encounter what is probably the first major divergence of thinking by way of information organization, which is the splitting away from organization by properties of language, such as alphabetical ordering, under the belief that all knowledge can be theoretically mapped in terms of concepts, and located by understanding its structure. In this way the DDC (Dewey Decimal Classification - dating from around 1880), UDC (extended from the DDC) and COLON (the most recent of the trio) have been born and believed in, causing information storage and retrieval-thanks to the explosion of recorded knowledge since

the Industrial Revolution - to become almost as important as the information itself.

The process of information handling was, by now, distinctly of two stages:-

(i) Interpret the information and store it with its means of access (a descriptive code)

(ii) Retrieve via the code.

At the same time as these developments, machines were making impacts in other business and administrative areas. The example towards which I am working is the machine conceived by Hollerith (c. 1880) for speeding up the derivation of census statistics by manipulation of punched-cards. By the 1940's, librarians were greedily seeking any machine able to lighten their load, and no distinction was made between the devices capable of offering assistance, be it the simple typewriter, or a sophisticated business machine dealing with punched cards - IBM was a common brand name of the time (QUIGLEY[5] (1941) and[226] (1952)).

An article entitled "More gadgets please "(MAMLAKIS[7] (1942)) sums up the feeling of the day, and doubts about substituting machines for "cheap labour" ("Guilds or Technocracy?" - QUIGLEY[8] (1933)) were swept aside by persuasive futuristic arguments (e.g. "Bibliographic robot next?" - COMPTON[9] (1937)), and there followed a glut of papers attempting to define the role of the machine amidst library surroundings (FAIR[10] (1936) and KOHLSTEDT[11] (1950)). An article by HAND[12] (1951) entitled "Special library of the future" attached as much importance to the colour scheme of a library as to potential for mechanization, whilst papers by HARDKOPF[13] (1951) and FAIRTHORNE[14] (1952) discussed the effect that Wiener's work on Cybenetics would have on library operation.

Library catalogues were obvious areas for using the fast improved punched-card sorting devices. In 1937, COMPTON[9,15] introduced the idea of selection of subject matter at a rate of 1000 titles/second mechanically, using a Hollerith-type, punched-card machine, whilst FRANCE[16] (1938) discussed a wide range of mechanical possibilities, ranging from microfilm to automatic catalogue production, and the taking of statistics to improve system performance. Details of practical punched-card systems began to abound (QUIGLEY[5] (1941) and FERRIS[6] (1948)) often having names limited by the lack of terminology, but none the less graphic for that (e.g. McCOY's "Bookometer"[17] (1940)).

Even at this stage, sorting on UDC codes was discussed, and it was felt that the unwieldy strings, though not precluding machine handling, could benefit from a little harmless manipulation. WUSTER[18] (1952) proposed replacement of symbols and commonly occuring codes such as 621.3 (Electrical Engineering) by more manageable letters. This technique was to be reborn some fifteen years later, when the UDC once more became popular in connection with machines. - more of this soon..

Details of a practical punched-card UDC-based retrieval systems were given in a paper by RUSTON[19] (1951) which, though in French, is more readable than some of the latter-day English publications on IR. The subject was also broached by VAROSSIEAU[20] (1948).

The burden taken on by machines also facilitated research into library practices, which had hitherto been obstructed by the sheer size of any document collection worthy of investigation. In 1962 SABEL[21] performed a statistical survey of the AWRE library at Aldermastan by recording data on punched cards. Of particular relevance to the present study, is the aspect of the AWRE project which concentrated on that part of the library organized by UDC. An attempt was made to compare the areas of the

catalogue given to heavy use, with those to which above average indexing effort had been applied. Figure 3.2 illustrates this point:-

	Subject breakdown of	
	Positive use of Information Office UDC catalogue	*Documents indexed/catalogued into Inf. UDC*
Chemistry and allied subjects	24%	15%
Health, safety, and allied subjects	12%	6%
Reactors and allied subjects	25%	20%
Metallurgy and ceramics	26%	17%
Physics and mathematics	8%	11%
Accelerators, electronics, controlled thermonuclear processes	12%	30%

figure 3.2- SABEL[21] (1962)

The major change brought about by machines for handling punched-cards was, however, in the development of freely post-coordinated or "keyword" indexing systems (of which TAUBE's Uniterm[22] (1955) was an orginal sophistication), that grew from methods based on optical coincidence of punched hole patterns, into the far-removed modern schemes of associative document retrieval. For further reference, an article by BLACK[23] (1962) reviewed the use of keywords, whilst JOLLEY[24] (1963) related "co-ordinate indexing" to other indexing methods. SCHULLER[25] (1960) claimed to have proven the superiority of indexing by Uniterms over other systems "for a collection of technical reports", whilst others[26] (1957) were more concerned with terminology, in deciding whether "coordinate indexing" was a better phrase than "aspect coordination". Sensible discussion was provided by PERRY[227] (1950), who stated some basic points about coordination (claiming that it provided more paths of access for machines) and the potential use of faceting. KENT[27] (1957) compared six index-term systems having various methods of operation (from hand-sorted to magnetic tape) and another paper by PERRY[29] (1954) attempted to deal with the theory of mechanized literature searching.

By far the greatest development to hit library thinking since the construction of the classical hierarchical classification schemes was yet to come however, and when it did come, the impact was no less profound than on the other business sciences. This was the computer of course, and after original diffidence, librarians came to appreciate that they had at their disposal a tool almost as powerful as the idealists amongst them had dreamt of. By 1960 the remnants of fantasy had been expunged, and computer applications in libraries were being put on a realistic footing. HEUMANN[30] (1960) wrote of a "Big Black Box at your beck and call", whilst others gave more soberly titled expositions on computer potential in libraries (IVALL[31] (1960), WARHEIT[32] (1960), NOLAN[33] (1959), McCORMICK[35] (1962), DONLEY[36] (1957) and ANON[34] (1961), as well as a 1966 review article by LIPETZ[37]).

Amidst the computer euphoria there was some doubt however; BOURNE[38] (1961), reviewing mechanized IR of all types (cards, tape search, computers etc.), over-estimated the future use of edge-punched-cards (which gave way to computers much faster than he expected), whereas MOOERS[39] (1960) said that ..." As computing machines are now designed, they are not matched to the job of information retrieval ..." and he foresaw a need for "computer-like ... information machines".

The first judgements as regards the suitability of the established hierarchical classifications (i.e. the UDC), was that they were not amenable to mechanization (Vickery, quoted by FREEMAN[40] (1964)) with the result that most effort was directed toward index-term methods (of little prestige prior to machine intervention). Inevitably, such methods have changed greatly, and given rise to a complex diversity of systems that belies their common origin. For example, the sophisticated and much documented SMART system due to SALTON[43] (1965), [42] (1971) and [41] (1973), is possibly the most advanced automatic working document retrieval system in existence (the degree of automation being high) and is one that has evolved - though by a long way - from an index-term basis.

To my knowledge, the only major work of any account that has been done on the mechanization of the classical, hierarchical classification schemes, is that of the mid - 60's, in which the UDC was found to be amenable to mechanization, though not with the efficacy of tailor-made schemes (which is to be expected). I shall return to this work in due course.

Nearing the present, it is reasonable to say that any library function likely to benefit from automation has been tried at some level, and certain tasks (notably catalogue production and maintainance) have been eased to such an extent, that many libraries depend heavily on their new-found ally. Not a library publication goes by without the enthusiastic ramblings of a newly - converted computer user effusing over the pages about his improved and labour-saving catalogue. Satisfactory developments in retrieval systems (and on-line systems particularly) are more rare however, and an emphasis on prototype rather than production becomes obvious when one scrutinizes the literature. Nevertheless, user needs tend to dictate to progress, and the solid establishment of certain hard-core working systems can only help to redress this imbalance. What is more, the wide spread net of current research, stretching as it does far off into the realms of statistical decision theory, must, in the long run, be for the good, especially for so young a subject.

The ability of machines, and more particularly computers, to take on time consuming and repetitive work of little intellectual content, is well known. A large amount of routine library maintainance, such as updating of catalogues &loan monitoring , has obvious parallels in established stock control problems. Evidence of the successful application of machines to these areas is plentiful.

What is more important however, is the fact that by dispensing with these burdensome tasks more efficiently, the computer is able to take

on other services that result from the flexibility with which one is now able to view a previously unmanageable collection. The most obvious examples of this are:-

(i) Catalogues can be produced in whatever format (physical or organizational) that suits the individual user.

(ii) Selective Dissemination of Information (SDI).

Evidence of (i) abounds (CAMPEY[44] (1972), MARKUSON[45] (1972), BECKER[46] (1970), BARUCH[47] (1966) and VERVLIET[48] (1974)). A paper by CHEN[49] (1973) documents the conversion of a catalogue, from Dewey to Library of Congress, with computer assistance.

At this stage, I should briefly mention the ever-growing area of machine usage that is an extension of the computer, eg. computer controlled phototypesetting and COM (Computer Output Microfilm). The use of microform materials in modern IR is well known (DRACHMAN[51] (1950), AVAKIAN[52] (1957), LEWIS[53] (1962), NEGUS[55] (1975) and ANON[54] (1964)) and the revolution arising from the refinement of COM to its present reliable and efficient state, is only a beginning. Extensive information on the latest hardware developments of interest to librarians is contained in the definitive Annual Review of Information Science and Technology (published by ASIS; e.g. [50] (1966)).

In talking of SDI under drudgery, one begins to undermine the intellectual aspect by which user profiles are constructed, since it is a technique owing as much to developments in retrieval , as to those facilities provided by computers for rapid searching and sorting upon which the more menial applications depend so heavily.

LYNCH[56] (1974) and [57] (1971) provides general and practical discussion on SDI, with particular reference to the Chemical Society information system, and CORBETT[58] (1970) discusses mechanized current awareness services at the AWRE. The UDC has been used successfully in various SDI systems (e.g. BECKER[59] (1968) -- see also SECTION THREE) and SCHNEIDER[60] (1971) has derived his own

hierarchical system (HICLASS) upon which to base SDI, which he obviously preferred to the "current keyword and co-ordination approach" in which effort is expended by every user, rather than at the classification stage. Schneider provides descriptive diagrams to illustrate his argument (see fig. 3.3) and goes on to say that the "value of an SDI system, or any IR system, may well be directly related to the amount of intellectual effort expended on indexing."

The aims of information retrieval have become more ambitious as improvements in computing power have been made. The early computerized systems were exclusively batch processes, and recourse to manual methods for urgent searches was a necessity. An early application of a digital computer to library searching was UNIVAC's FACTRONIC system - publicized by MITCHELL[61] (1953) - capable of selecting one item from one million (based on 15 criteria) in four hours.

As interactive usage become a reality however, the potential benefits of on-line retrieval systems were soon recognised, and in designing for exclusively mechanized use it became possible to ignore those development criteria which provided for a high degree of manual operation. Today therefore, the cream of the retrieval systems are accessible interactively (although batch access is often provided as a cheap alternative) but it must be immediately recognised that these computer orientated hybrids are subject to a blight which did not trouble the manual systems, that is, mechanical failure. Emergency back-up for a computer-based system can take one of three forms:-

(i) A second computer (or more?)

(ii) A manual version (eg. microfilm).

(iii) Wait until repaired (this is a serious alternative, though not one I shall consider further).

It was mentioned earlier however (and will be expanded in the remainder of this section) that the systems most enthusiastically developed in the

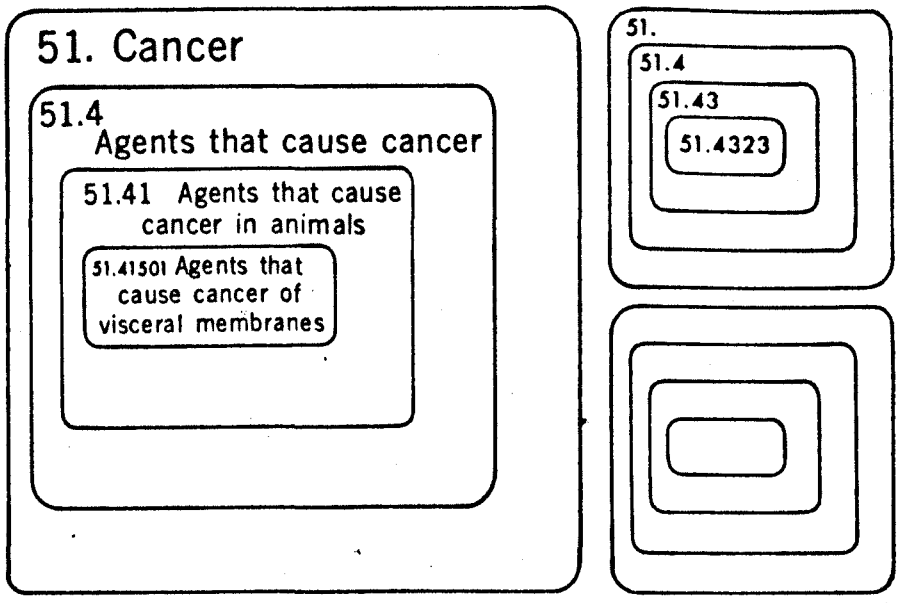


Diagram of the linear, general-to-specific logic used to index and retrieve information in HICLASS systems based on enumerative hierarchical classifications. The multiple sets of circles represent different, integral concepts in an article on the causation of mesotheliomas by asbestos in mice.

figure 3.3 - SCHNEIDER[60] (1971)

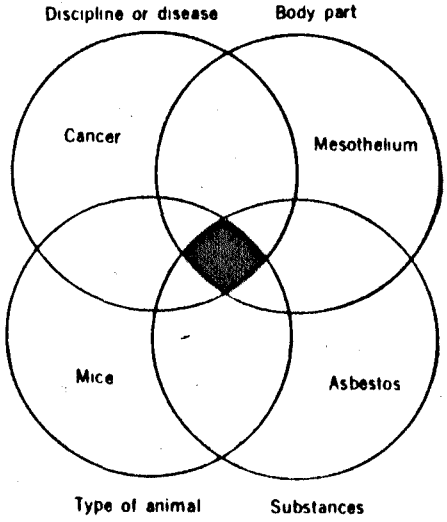


Diagram of coordination logic required when information is broken into keywords or other isolated terms during indexing and is then recombined during retrieval. This is an alternative way of indexing the article

rush of automation were, and are, those systems that were not feasible before machines to operate them became available. Such systems cannot, therefore, perform anything like as well (in fact they usually don't perform at all) when only a manual alternative is available. In this case, it is fair to say then, that undiminished reliability can only be ensured by duplication of machinery (the most expensive alternative).

With computers at their disposal, library researchers who had long repressed their suspicions in the inadequacies of the classical hierarchical classification and retrieval systems, found a justification to let their minds range over the new potentialities. The immediate reaction was to investigate the various "keyword" systems that had enjoyed a certain popularity when operated by more obviously mechanical means (hole coincidence) and for which the computer logic of Boolean operation seemed to be so perfectly designed.

The idea was that a "controlled vocabulary" (a set of index-terms or descriptors) could be used to coordinate (describe) a document in a theoretical space, having as many dimensions as the vocabulary had terms. The main differences between the many early keyword systems were by way of the means of selection of the terms. Should they be extracted from the title, the abstract, the whole document, or should they be fixed by an omniscient classifier?

A bibliography on coordinate indexing is provided by ERSKINE[62] (1963) and a later case study of the Uniterm index by MATTHEWS[63] (1968), but in general, doubt was spreading as to the future of keyword systems. For example, LANE[64] (1964) said that "there is serious doubt as to the universal applicability of KWIC (keyword in context) indexing", which

was something of an understatement to any but the most ardent supporters of Keywords. A degree of semantic control was required, and in a number of cases this was achieved by harnessing the services of the UDC, VERVLIET[48] (1974), VAN HALM[65] (1972), FREEMAN[66] (1967) and SCHULLER[67] (1960).

Divergence between keyword systems began when the limitations (not to say errors) imposed by a fixed and unyielding term vocabulary became obvious. Before expanding on this, it should be said that by now (early 60's) reasoned criticism, based on thought-out system evaluation, was being used to direct the flow of development. The almost legendary Cranfield project, conducted under the auspices of ASLIB and more particulary Cyril CLEVERDON et al[68] (1962 and 1966), had put retrieval system evaluation on the map, and performance measures such as Precision and Recall become recognised descriptors of system efficiency. In Cranfield II, Cleverdon wrote "... quite the most astonishing and seemingly inexplicable conclusion that arises from the project, is that single term index languages are superior to any other type" (see fig 3.4).

Order of effectiveness of three types of indexing languages. [Adapted from Cleverdon and Keen

Type of indexing language	Rank orders for methods using indexing language	Average score for language
Single terms: content words manually chosen from full document	1, 2, 3, 4, 5, 6, 7, 12	64.15
Controlled terms: single terms modified by look-up in manually constructed thesaurus or authority list	10, 11, 15, 17, 18, 19	60.34
Simple concepts: single terms concatenated into standard noun phrases reflective of document content	8, 9, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33	54.55

figure 3.4 - SALTON[69] (1970)

This conclusion, which is no less strange than Cleverdon himself realised, has been criticized, not least on the grounds of the inadequacies of the test collection (1400 documents in the limited subject field of aeronautics), but SALTON[69] (1970) has supported this result with experiments

performed on his SMART system. Salton showed that certain refinements, expected to be advantageous (term hierarchies and syntactic phrase analysis), carried no benefits, whilst term weighting and synonym control were found to produce marked improvements. In fact VICKERY[70] (1960) had already recognized the importance of thesauri, and REES[71] (1962) argued so strongly for a "coded thesaurus" to even out time fluctuations in the languare, that one wonders if he had ever heard of the UDC? Suffice it to say that by the time Salton had practically consolidated the arguments for synonyn control any keyword type system lacking this feature was felt to be not worthwhile.

With this, came the understanding that new forms of analysis were bound to be instrumental in making dramatic improvements in the efficiency of retrieval systems. Also, increased attention to methods of evaluation ensured that streamlining of effort was a natural consequitor of the new advances. By the mid - 60's sophistication of keyword methods had occurred to such an extent that they bore more relation to the infant systems based on statistical decision theory (see later in this section) than to the index-term schemes of the Uniterm genre.

For a start, automatic derivation of index-term lists (a form of automatic classification) was becoming popular, leaving the computer to "read" the document (or part of it) and build its vocabulary accordingly; a development that PERRY[227] (1950) had not expected only a decade earlier. The major exponent of such all pervading automation was SALTON using SMART, which has a seemingly endless number of modes available with which to perform retrieval (see SALTON[42] (1971)) in the hope of replacing simple word matching by "intellectual aids" (SALTON[69] (1970)). What is more, in the course of developing SMART, Salton has isolated problems of a more general nature in IR, such as the need to make up for missing language analysis by automatic means (SALTON[228] (1973)).

Development along these lines has incorporated the use of "associative document retrieval", in which the document is associated to the term vocabulary with a greater complexity than was the case with the limited ON-OFF term flagging of the earlier systems (SALTON[72] (1963)). This is achieved by an association matrix e.g. :-

		DOCUMENTS				
		1	2	3	.....	N
	1 -	x_{11}	x_{12}	x_{13}	.....	x_{1N}
	2 -	x_{21}	x_{22}	x_{23}	.....	x_{2N}
TERMS						
	3 -	x_{31}	x_{32}	x_{33}	.....	x_{3N}
	⋮	⋮	⋮	⋮		⋮
	n -	x_{n1}	x_{n2}	x_{n3}		x_{nN}

where xij denotes the degree of association between a given document and a given term ($0 \leq xij \leq 1$). In retrieval the query takes the form of an additional column, which is compared with the DOCUMENT columns to detect matches. Other matrices can provide equally useful association information : a Document × Document matrix can help identify interrelations within a collection, whilst Term × Term matrices have been used to tighten vocabulary control and improve retrieval (STILES[74] (1961) and JACQUESSON[73] (1973)).

Fully automated retrieval methods have developed in many ways. SALTON[72] (1963) proposed the use of citations as a potential aid in highlighting connections between documents and, in a more latent sense, the descriptive terms by which they are defined (thereby permitting consolidation of the thesaurus as new terms arise). Also, it is possible to select sentences having a high frequency of index-terms, and also sentences containing

index-terms of high information value (i.e. rarely used terms) thus performing "auto-abstracting" when the sentences are combined (though with what comprehensibility I do not know). Auto-abstraction is a form of "fact distillation", and in this guise can form a basis for question/answer systems (e.g. O'CONNOR[2] (1973)).

The other major class of document retrieval system to rise from the ashes of the keyword is less advanced (as regards degree of automation) but certainly more widely used, thanks to its being cheaper. In essence, this type of system is a keyword system, augmented by a thesaurus, and perhaps by other aids which help avoid semantic traps and may, in some aspects, stray into the territory of the fully-automatic methods. The important feature of such systems however, is the additional facility for text-searching (often limited to the abstract) in which case one avoids specially applied index-terms and interacts with the raw document (see SWANSON[75] (1960)). It should be stressed that such systems only remain cheap when used sensibly, and without over-complicating the search logic. It is also to be noted that this technique works on the opposite principle to the associative retrieval systems, since a text search does not require any form of classification. A little manual effort expended in providing loose index-term descriptors enables the number of eligible documents to be reduced as a preliminary step, but the general idea of "classification by query" is the feature to bear in mind.

A notable example of such a system is Lockheed's DIALOG which, though based in California, is available throughout Europe, and was used during the course of this survey. DIALOG is a reasonably sophisticated system, and a recent costing study by ELMAN[76] (1975) proves it "efficient and cost-effective". A similar system is discussed by HALL[77] (1971).

A bibliography of working document retrieval systems (based however loosely upon index-term vocabularies, and largely of the latterly discussed type) is contained in a most readable book by LANCASTER & FAYEN[78] (1973), which includes brief details of such systems as DIALOG, ORBIT, STAIRS, SMART, BROWSER etc. Others such systems include BOLD & MEDLARS (BLACK[79] (1966)), the latter being the batch fore-runner on the on-line medical IR system MEDLINE; ORBIT (MORROW[81] (1976)) - a paper containing some fetching flow-chart diagrams); SOLAR (MITCHELL[80] (1973)); MOLDS (ATHERTON[89] (1970)) important for its use of the MARC tapes; CAIN (VAN DYKE[90] (1972)) and ASSASIN (CLOUGH[82] (1971)). More general articles are provided by BACK[83] (1972), BORMAN[84] (1972) and by WESSEL[85] (1975), whose book adopts an emimently practical approach. Non-specific reviews appear in the books of DOYLE[86] (1975) and HENLEY[87] (1970), and MYERS[88] (1973) has reviewed the use of computers in searching law texts.

Compared with the associative document retrieval techniques, the third class of system that I am about to discuss would seem to lack novelty. Such a denigration would be unfair however, since work on document indexing by Cluster Analysis has been progressing for some years, and has arrived at its present state by taking quite different views of IR problems. The main protagonist of "clustering" in this country is Karen SPARCK-JONES, and she has contributed a number of papers on the subject ([91] (1964), [92] (1970), and [93] (1973)) and fired enthusiasm in others (e.g. WOLFF-TERROINE[94] (1971) and FIELD[95] (1975)). The idea is to computer analyse documents for words of information value (index-terms as it were) and thereby position the document in a concept space, weighted to reflect the high information content of certain terms. Classes of documents can then be defined by cluster analysis. Retrieval is a matter of relating the point in space, produced by analysing the query, to adjacent "clumps" or classes. Certain parallels with associative document retrieval should be obvious (a recent paper by SALTON[126] (1975) confirms this), but the statisticians' involvement stresses the role of the mathematician in the expanding field of Information Science (even if working

document retrieval systems based on clustering do not abound).

As regards other statistical methods, back in 1952 SHERA[96] noted the applicability of decision theory to IR, and BAKER[97] (1962) suggested using latent class analysis for the classification and retrieval of library data. An interesting and wide-ranging system due to MARON and KHUNS[98] (1960) caused a statistical inference to be made as to the relevance of every document in the collection to a given request (the "relevance number") and the need for such optimizing approaches was later appreciated by TRITSCHLER[99] (1964).

Before passing on, a word should be said for the theoretical conjectures that have possibly influenced the development of working document retrieval systems, without necessarily providing practical support. Books by BECKER and HAYES[100] (1963) and BOURNE[101] (1966) provided an early basis for the growth of mechanized information handling, and have been followed by JAHODA[102] (1970), MITCHELL[103] (1971), VICKERY[104] (1973), BECKER and HAYES[46] (1970), DOYLE[86] (1975) and, in a book of exclusively theoretical content, KOCHEN[105] (1974), whose contribution is unique.

Reviews by LAWLOR[106] (1962), WALSTON[107] (1965), JACKSON[108] (1971) and CLEVERDON[109] (1970) point to articles emanating from diverse researchers, and the Annual Review of Information Science and Technology monitors important theoretical innovations. Individual papers are myriad, and vary greatly in readability. A ten-page criticism by BAR-HILLEL[110] (1957) is remarkable by virtue of the fact that at no time is it constructive, but its audacity has ensured continued citation (e.g. O'CONNOR[111] (1964)).

Two important implementations deserve mention here. The first is PRECIS, which is a widely used computer-based method for catalogue production (the British National Bibliography for example) and, being

based on a manually applied, structured indexing system, occupies a unique position; it is largely the brainchild of Dereck AUSTIN[113] (1974) and [114] (1974). The emergence of a national-scale library operation, relying upon a completely computer-based system (using COM, computer typesetting etc.), must be considered a major advance.

Secondly, there is the internationally known MARC project, which has proven successful in defining a standard format for recording book information, thus facilitating world-wide circulation. Centrally produced MARC tapes have made large scale machine processing of library information possible, without which, such systems as PRECIS (which interfaces with MARC) would have been slow to advance. A paper by ATHERTON[89] (1970) describes a retrieval system based on the MARC tapes, and VERVLIET[48] (1974) has documented the installation of a MARC-compatible cataloguing system.

All computer-based document retrieval systems rely upon certain general developments in both computer hardward and software. For example, many papers are given over to file organization and search strategies, such as GUTHRIE[115] (1972), COSTELLO[116] (1962), VAN HALM[65] (1972), DAVIS[117] (1972), BOOKSTEIN[118] (1974) and reviews ANON[119] (1966) and [120] (1972). KEEN[121] (1968) has compared search strategies in manual and automated systems. An extension of search strategy is ranking of search output in relevance to the user, and this is taken up by MILLER[122] (1971), DYKE[123] (1959), MARON[98] (1960) and TRITSCHLER[99] (1964).

Again in connection with search strategies, the need to appreciate that different words have different "information values" has been recognised by ROBERTSON[124] (1974) and BELZER[125] (1971). For example, a word used in only one document of a large collection is immensely valuable, as retrieval could then be based on that word alone; this

is illustrated by SALTON[126] (1975) and by fig 3.5. Implementing such considerations, by means of term-weighting, has been shown by SPARCK-JONES[127] (1972) to give notable improvements (see fig 3.6).

Terms in Discrimination Value Order

(1963 TIME magazine)

GOOD TERMS	POOR TERMS
1 Buddhist	7560 Work
2 Diem	7561 Lead
3 Lao	7562 Red
4 Arab	7563 Minister
5 Viet	7564 Nation
6 Kurd	7565 Party
7 Wilson	7566 Commune
8 Baath	7567 U S
9 Park	7568 Govern
10 Nenni	7569 New

figure 3.5 - SALTON[126] (1975)

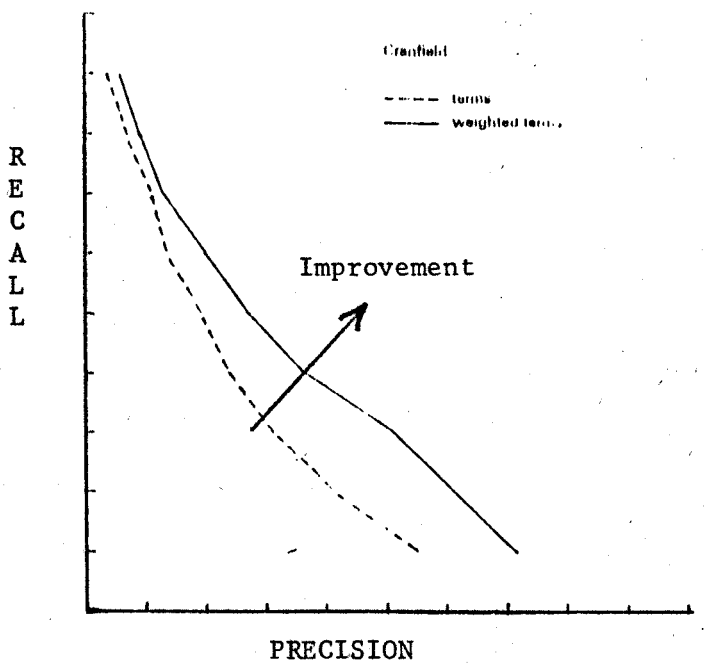
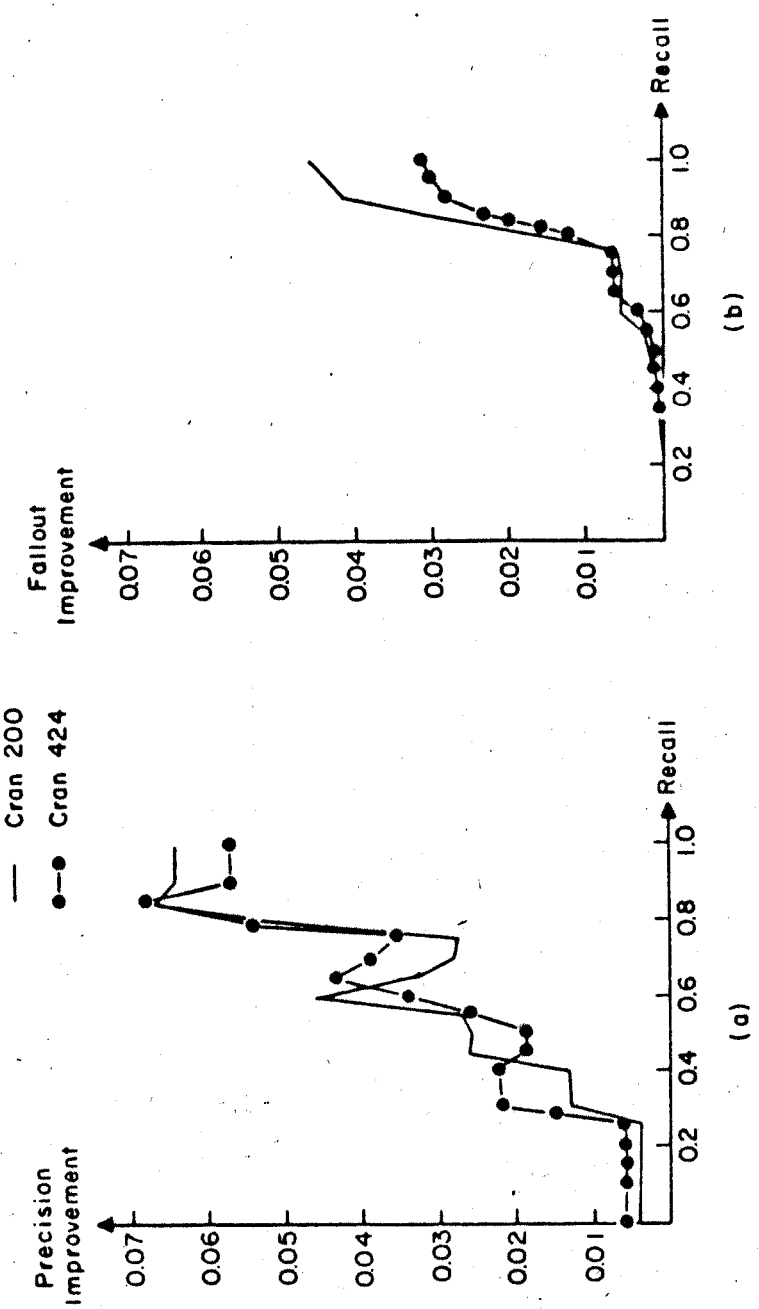


figure 3.6 - SPARCK-JONES[127] (1972)

Also, new practices for operating document retrieval systems can often cross the narrowing boundaries between diverse techniques. SALTON[128] (1972) for example, has long expounded the virtue of involving the user with the search interactively, to improve performance by "feedback", and he has published convincing proof of this (fig 3.7). JACQUESSON[73] (1973) has found the same, and YU[129] (1976) has solicited feedback from the user as to the relevance with which he feels his request to have been answered, in order to dynamically tailor the search strategy ("relevance feedback").



Precision (a) and Fallout (b) Improvement for Feedback Strategy 1 (positive feedback) (averages over 22 queries)

figure 3.7 - SALTON[128] (1972)

A paper by THOMPSOM[130] (1971) describes a retrieval system in which hierarchies are displayed on a cathode ray tube. The user selects terms by means of a light pen, and his "interaction vocabulary" may be expanded by at least an order of magnitude (it is claimed). In Thompson's words ... "A hierarchical structure was chosen because it seems to replicate the structure of cognitive thought processes most closely", thus allowing the simplest transfer of the problem into the structure and vocabulary of the system (a view held by UDC arbiters since time immemorial).

SPARCK-JONES[131] (1973) has proposed a method for characterizing the separation between relevant and non-relevant documents in answering a query, in order to gauge the need for, and effectiveness of, proposed modifications; whilst a theoretical paper by MEINCKE & ATHERTON[132] (1976), refreshingly returns to basics, and assesses classification and indexing systems from the elementary standpoint of concept space. This article contains a table due to Vickery, which lists methods of classification and indexing arranged by increasing order of control (reproduced here as figure 3.8).

Existing Methods of Classification and Indexing
(arranged by increasing degree of control)

1. Words chosen from title or text, with common words omitted
2. Words chosen from text, with omission of common words and *consideration of variants*.
3. Words chosen from text, with omission of common words, consideration of variants, and *generic relationships*
4. Words chosen from text, with consideration of *syntactical relationships between indexing terms*.
5. Any of the preceding methods, with *addition of terms not used in text*.
6. Assignment of index entries from a fixed authority list or classification scheme.
7. Assignment of index entries from authority lists or classification schemes representative of *several viewpoints* and *aspects of subject*.

figure 3.8 - MEINCKE & ATHERTON[132] (1976)

The important subject of user-training is taken up by MOGHDAM[135] (1975) who prefers the use of CAI (Computer Aided Instruction) in conjunction with the on-line retrieval system being taught.

Though not completed, the bridge between computer programmers and information specialists is certainly begun in a book by MEADOW[136] (1967) and articles by DOLBY[137] (1971) and BATTEN[229] (1967).

Finally, no discussion of general mechanized IR would be complete without mentioning databases, of which WILLIAMS[225] (1974) and ANON[119] (1966) give details, and CUADRA[138] (1971) praise.

Automation of the UDC

When computers first become available to library researchers, it was not spite that caused them to neglect the automation potential of the classical classification schemes, but rather the format restrictions imposed by the early machines. For this reason, variable length strings of digits, characters and symbols were considered sufficient justification for leaving the UDC alone. By the time these early problems had been overcome, advances in other systems had occurred at such a rate (by the mid - 60's SMART was well advanced) that automation as applied to the hierarchical classifications was comparatively non-existent.

The UDC was felt to be better suited to machine searching than either the Dewey or L of C Schemes since, according to HINES[139] (1967) ... "it was not designed primarily for ordering documents on shelves

to provide a single main aspect approach" - a feeling supported practically by HANSEN[140] (1968); and, in a manual sense, CLEVERDON[141] (1960) gave it the stamp of his approval, in saying that a UDC catalogue can be used effectively by "persons with little or no previous experience". Others claimed UDC superiority over Uniterms (TELL[142] (1969) and MILLS[143] (1964)), and although classificatory criticisms (WAHLIN[144] (1963) and MAROSI[145] (1969)) and suggested improvements (ARNOLD[146] (1958) and PERREAULT[147] (1968)) were not in short supply, the overwhelming need to attempt mechanization for the sake of the large existing UDC-based collections, was predominant.

Gradually, the feasibility of sorting UDC strings was appreciated and. as was the usual first step, automated catalogue production and maintainance was undertaken. There were early problems of course; UDC strings, though manageable, were far from ideal, but it was in coping with early difficulties posed by the symbol-ridden strings, that much of the groundwork for code manipulation, and hence retrieval, was carried out. Notable examples of UDC catalogue maintainance include those systems used by AYRES[148] (1967) at the AWRE, SAVILLE[149] (1964) at the Iron and Steel Institute, PATTEN[150] (1974 at British Steel, NEVILLE[151] (1975) at the Buildings Research Establishment, MAROSI[152] (1969) at the Euratom centre and VERVLIET[48] (1974) at Antwerp library. Work by FREEMAN[66] (1967) and RUSSELL & FREEMAN[153] (1967), stresses the latters contribution to UDC mechanization. VAN HALM[65] (1972) has used short UDC numbers (12 digits) for catalogue production.

Work on developing a mechanized UDC retrieval system was such an obvious next step, that no one person can be attributed with its commencement - although RUSTON[19] has used the UDC for retrieval in a punched-card system as long ago as 1951. However, UDC-based retrieval posed difficulties, and

there was no shortage of doubt concerning a successful outcome. VICKERY[40] was not enthusiastic but, to be fair, his aspersions are taken from a time when computers were less flexible than they later became, whereas LESLIE[154] (1961) rejected the UDC as being too strict and unyielding (though working in a narrow subject area he might have expected this). Only recently, SALTON[41] (1973) has said that "simple duplication by automatic means, of standard manual document indexing and retrieval operations will not produce acceptable results", which is a tacit indictment of UDC. In extolling the virtues of Precis, AUSTIN[113] (1974) has also condemned the UDC (though conceding its superiority over non-faceted systems) in anything other than the single-field collections in which computerized UDC retrieval has been proved successful, whilst GOLD[155] (1972), to redress the balance, has criticized Precis as being inferior to a system that can combine subject indexing with subject classification, such as the UDC.

On the constructive side, PERREAULT[147] (1968) and DAHLBERG[156] (1971) have suggested modifications to the UDC to increase its suitability for machine handling, and FREEMAN[40] (1964) clearly identified the light at the end of the tunnel for computer retrieval based solely on UDC strings.

In general, programs for simple matching of strings were augmented by an understanding of those elements in UDC numbers worthy of isolation. The early work of WÜSTER[18] (1952) has been discussed already; since then, CALESS & KIRK[157] (1967) have described a method to meaningfully split UDC strings, without offending the "delicate relationships built into the assigned number in the classification process". This was done by grouping combinations of (a) main class numbers, (b) main class numbers with auxiliaries and (c) auxiliaries with auxiliaries, in accordance with the intent of the assigner of the original UDC number. In fact, all UDC retrieval systems require some harmless string manipulation, but that described by CALESS & KIRK is the most detailed of all that is published.

Before long (1966) work on systems either wholly or partly dependent upon mechanized UDC retrieval was more widespread than at any time before or since. McCASH & CARMICHAEL[158] (1970) developed SDI services for the Iron and Steel industry based on UDC user profiles, and achieved success with greater than 95% Relevance (which is good), whereas on SDI system run by BECKER[59] (1968), using 12 digit UDC codes, led to an increase in output by 40% and a cost reduction of 25%. CALESS & KIRK's[157] retrieval system for seismology, was justified by the statement that ... "By machine searching, the advantages of free access to all concepts - which is the great virtue of post-coordinated systems - is combined with the considerable advantages inherent in the highly structured vocabulary found in the UDC".

It will be noticed, that in the above examples, as in many document retrieval systems, the range of subject matter is often limited to a specific application, which is certainly not the case at the BBC Film library. It is usually to be expected that the UDC will operate more efficiently when covering a wide subject area, since finer and finer splitting of the decimal notation - to accomodate repeated subdivisions - is then less imposing. Workers in highly specific subject areas may plagiarise UDC strings by removing digits from the front, thereby disposing of unnecessary generality.

By far the most extensive work on mechanization of the UDC for retrieval, has been that performed by FREEMAN & ATHERTON et al[224] (1968) in the USA, under the title of "The American Institute of Physics UDC Project" (AIP/UDC) which has resulted in a set of detailed reports, and the conclusion that "the UDC can be successfully mechanized" - a conclusion (thankfully) supported by extensive practical work.

In "AIP/UDC no.6", Freeman, operating a computer system on a metallurgical document collection, stressed the importance of designing UDC search strategies by taking into consideration "actual distributions and combinatorial properties of the UDC numbers used for indexing". The most significant acheivement to come out of the AIP/UDC project was however, the production of an on-line document retrieval system called AUDACIOUS (operating on a Nuclear Science collection), based wholly upon the UDC at both the algorithmic and interactive levels, that is, both the machine and the user were directly involved with the handling of UDC strings. An earlier paper by FREEMAN & ATHERTON[159] (1967) gives details of UDC number encoding (see fig 3.9) and "AIP/UDC no. 7" contains a brief operating manual for AUDACIOUS, and many sample runstreams that illustrate additional aspects (use of logic etc.)

In a working system, the need for the user (if he is a layman) to deal with the UDC strings at all, could perhaps be questioned. Direct transfer from supplying the original term, to the commencement of a search, would doubtless be possible if a degree of control could be forfeited (optionally) by routing through a machine stored subject-headings index.

At this prototype level however, the feasibility was on trial, and not the potential for endless sophistication and refinement. Although the system was not subjected to rigorous testing (evaluation was even more primitive in the 60's than it is now), there would seem to be no obvious reason for attributing to it a greater inefficiency than to any other document retrieval system at an early stage of development.

The advantages that one might have expected to gain by mechanizing a hierarchical system were not lacking, notably the ready-made provision of hierarchies through which to ascend and descend, and one might assume that browsing outside the boundaries specified in the original search

UDC INDEX NUMBERS ENCODED FOR STORAGE AND RETRIEVAL BY COMPUTER

Type	Name	Normal Form	Encoded Form	Example	Encoded Form
Content Facet	General Subject	n	Cn	551.524.63	C55152463
Form Facet	Language	=n	En	=30	E30
Form Facet	Form of Work	(0n)	Fn	(084.3)	F843
Content Facet	Place	(mn)	Pmn	(265)	P265
Content Facet	Race	(=n)	Rn	(=30)	R30
Content Facet	Time	"n"	Tn	"475"	T475
Subordinate Content Facet	Point of View	.00n	Vn	55.002.2	C55V22
Subordinate Content Facet	Special Auxiliary	-n	Wn	62-451	C62W451
Subordinate Content Facet	Special Auxiliary	.0n	Xn	62.018.7	C62X187
Connective	Synthetic Connective	n'n	nYn	546.32'13	C54632Y13
Connective	Inclusive Connective	n/n		543/546	C543
Connective	Relative Connective	n:n		543:546	C546
Connective	General Connective	n+n		543+546	(Connectives not encoded)
Connective	Subordinate Connective	n(n)		543(546)	

Note: n = a set of digits, any of which may be any of the digits, 0...9.
m = a digit from 1 to 9.

figure 3.9 - FREEMAN & ATHERTON[159] (1967)

would pose few difficulties (not so in other document retrieval systems).

What must be assumed to be the major impact of this project however, is to be measured in the relief felt by heads of UDC libraries the world over, assured that an alternative exists between, on the one hand, eternal manual operation and, on the other, translation of a UDC catalogue to some other system permitting computerization.

To burst the balloon briefly, "AIP/UDC no.8" isolated reasons for search failures, and found the most common cause to be vagaries of query formulation - search analysts (the counterparts of Enquiry Assistants at the BBC Film Library) differing somewhat in their strategies. We return to optimism however, with a paper by de REGT[160] (1968) presented at the euphoric FID-sponsored Copenhagen conference on "The UDC in a Mechanized Retrieval System" (the FID administers the UDC - hence the euphoria), in which the conclusions of "AIP/UDC no. 9" are recorded, and are reproduced here as fig 3.10. The conference concluded however, by saying that "a strong need was recognized for further investigation, especially in very large databases.

Conclusions of the AIP/UDC project

The researchers Freeman and Atherton in their final report (Report AIP/UDC-9) came to the following conclusions:

- There is no longer any doubt that the UDC can be used as the indexing language in a mechanized system. No barriers exist to the successful use of the UDC in either a batch-processing or an interactive mode.
- The results of the project should lend support and encouragement to those who consider using of UDC in computer based retrieval systems.
- No insoluble problems were found, but the long-existent matter of the theory according to which the UDC will be developed in the future is seen to be accentuated by the requirements and capabilities of computer-based systems.
- On the basis of experiments in a test environment which reasonably simulates a real information system, it is felt justified to encourage those who wish to make use of UDC as the indexing language in a computer-based retrieval system.

figure 3.10 - de REGT[160] (1968)

In fact, following the Copenhagen conference, enthusiasm for the UDC was reborn, notably in its ability to overcome language barriers. SAMUELSON[230] (1971) makes this point, adding that UDC re-classification (if it were to be the case that mechanization proved impossible) could be ruinous, and a review article by BRANDHORST & ECKERT[162] (1972) charts renewed interest in the UDC due to "waning" of the thesaurus period, and records UNISIST's recommendation for "a continuing program to strengthen the UDC, and further studies to test its applicability to retrieval systems". LLOYD[163] (1969), in talking about the UDC in its international aspects, has said that "use of the UDC in mechanized IR systems has been really put on the map ... mainly in the USA".

Since the end of the AIP/UDC project, published details of mechanized UDC retrieval systems are simply non-existent. A number of automatic document retrieval systems use the UDC in one guise or another, but seemingly not as the central code upon which to base retrieval. I feel that this is unlikely to be due to the exhaustivity of the AIP/UDC project, since the acceptance of Freeman & Atherton's findings should have led to work, and work to publication; but rather to the continuing latent suspicion that inhibits librarians from using the words "UDC" and "mechanization for retrieval" in the same breath.

Evaluation of IR systems

The theoretical side of evaluation of retrieval systems rests largely upon a small number of classical papers, and the discussion which has followed some pioneering work of CLEVERDON et al[68] (1962 - 1966), known generically as the "ASLIB Cranfield Project". The practical side on the other hand, relies heavily upon the provision of manageable test collections - the properties of which are well known - and the formulation of queries that are chosen to be as much devious as representative.

These two generalizations (which may not be strict facts) perhaps explain why universally reliable efficiency evaluations do not exist (which is a fact). There is another reason however, and it lies in the confusion between Efficacy & Efficiency. The acid test for a retrieval system is that it produces good results with ease (rather than with a performance characteristic of a certain shape) and so it is that the "systems view" of an IR process can often be beneficial. The rather loose theoretical aspect of "systems thinking" (in this guise at least), is rather troubling in a subject eager to gain "hard" scientific status (O'CONNOR[2] (1973)), and so theoreticians are given rather a lot of rope with which to try and make the transition more stable.

In 1963, SWETS[164] gave an in-depth treatment of retrieval system performance in the light of decision theory (he cites the work of MARON & KUHNS[98] (1960) that was mentioned earlier) and poses his argument with an elegance that has ensured continued respect for his work (e.g. BROOKES[165] (1968) & HEINE[166] (1974)).

It was Cleverdon's undertakings however, that caused the major impact in the early 60's, thanks mainly to its intensely practical nature (which was something of an innovation at the time). The results of the Cranfield project are well documented, both by CLEVERDON[167] (1963), [168] (1967) and [68] (1962 - 1966) and others, including SALTON[69] (1970), SWANSON[169] (1965) and KYLE[170] (1964). Briefly then, Cranfield I ("Report on testing and analysis of an investigation into the comparative efficiency of indexing systems") found equality between four systems - a faceted classification, the UDC, on alphabetical subject index and a Uniterm index - in that all four systems achieved recall of between 75 and 85% (classifier mistakes being the main causes of error), by applying 1,200 questions to a collection of 18,000 documents in areas related to aeronautics. The disputed conclusion of Cranfield II ("Factors determining the performance of indexing systems")

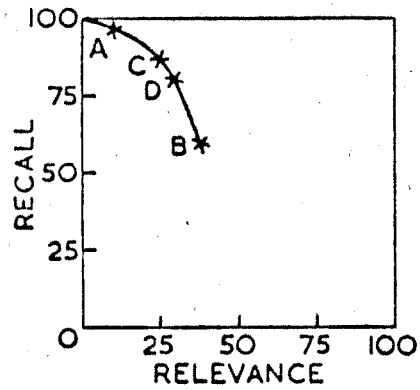
which claimed superiority for single index term languages over "any other type" will not be discussed further: recent changes in IR, brought about by continuous accelerations in computing power, have inveigled alternatives that the Cranfield workers were unaware of, and conclusions drawn in the past (largely in a manual sense) need not apply today.

The lasting contribution of Cleverdon however, is to be seen in his use of RECALL & PRECISION (RELEVANCE) and, more notably still, in the relationship between RECALL & PRECISION as a measure of system performance. To cope with the definitions:-

$$\text{RECALL} = \frac{\text{No. relevant docs. retrieved}}{\text{No. relevant docs. in collection}} \times 100$$

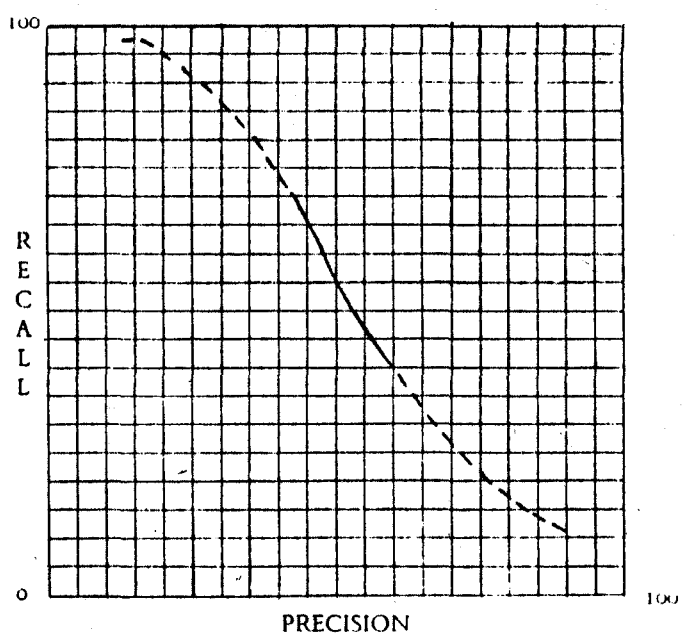
$$\text{PRECISION (RELEVANCE)} = \frac{\text{No. relevant docs. retrieved}}{\text{Total No. of docs. retrieved}} \times 100$$

Plotting Recall against Precision yields a "performance curve" in a "Precision - Recall square", and such a curve implies (according to CLEVERDON[171] (1904)) that there is an unavoidable trade-off between Recall & Precision in an IR system. Fig 3.11 for example, is one such Precision-Recall square (Cleverdon's comments on it are included for completeness) whereas fig 3.12 is the theoretical completion of such a graph, taken from a later paper by CLEVERDON[172] (1972), in which he writes ... "in a large number of situations, an improvement in recall can only be obtained with a loss of precision, or vice versa, and it is reasonable to operate a system using this as a working principle". Discussion and reviews of system evaluation at this basic level are provided by CLEVERDON[173] (1970), REES[174] (1966), BOURNE[175] (1966), KREVITT[176] (1973), FAIRTHORNE[177] (1965), KENT[178] (1955), KATTER[179] (1969) and JACKSON[108] (1971).



Referring back to the graph, it may be shown that a retrieval system that employs a specific vocabulary and practises exhaustive indexing is capable of operating anywhere on the maximum performance curve between A and B *by means of variations in search programmes*. But a retrieval system employing a nonspecific vocabulary and nonexhaustive indexing will only be able to operate between the points C and D and will never be able to improve on this performance *however much the search programmes are varied*.

figure 3.11 - CLEVERDON[171] (1964)

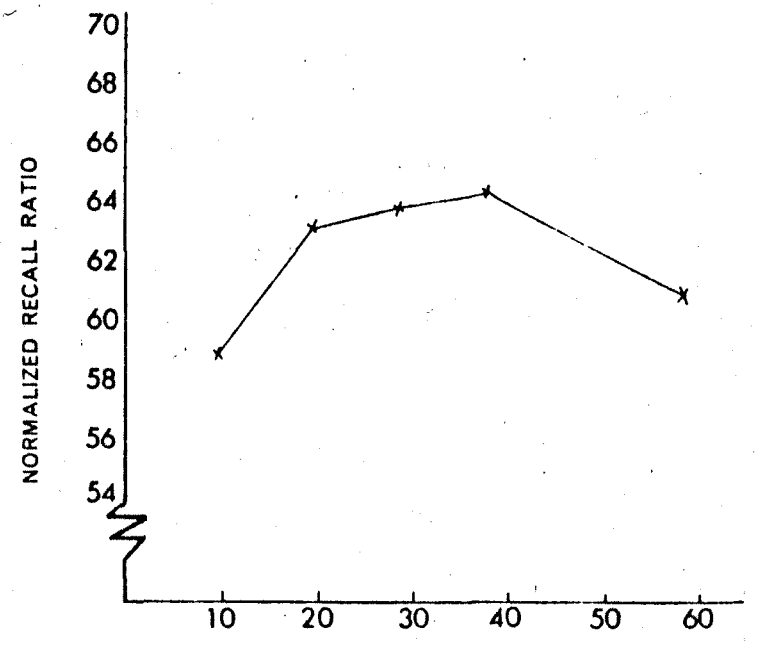


Estimated performance curve of MEDLARS (1967)

figure 3.12 - CLEVERDON[172] (1972)

Cleverdon's free use of recall and precision, together with his hypothesis concerning their inverse relationship one to the other, began an avalanche of papers concerning these and other simple performance measures. SWETS' more intricate theories are still referred to, and obviously respected, but the ease of assessing recall etc., makes these last yardsticks a more common choice.

CLEVERDON again[168] (1967), investigated the relation between recall and "exhaustivity of indexing" (fig 3.13), whilst ROBERTSON[180] (1969) considered FALLOUT - i.e. the probability of retrieving a non-pertinent item (SWET's "false drop") - to be a better measure of performance (along with recall) than precision (see fig 3.7).



PLOT OF NORMALIZED RECALL RATIOS FOR FIVE LEVELS OF EXHAUSTIVITY OF INDEXING

figure 3.13 - CLEVERDON[168] (1967)

TAUBE[181] (1965) was more scathing in his view of Cleverdon's work, saying ... "It follows that the ... claim of the Cranfield studies to have discovered a mathematical ratio that will permit precise evaluation of systems ... (is) without substance or merit" and he has criticized the alleged "pseudo-maths" of it all.

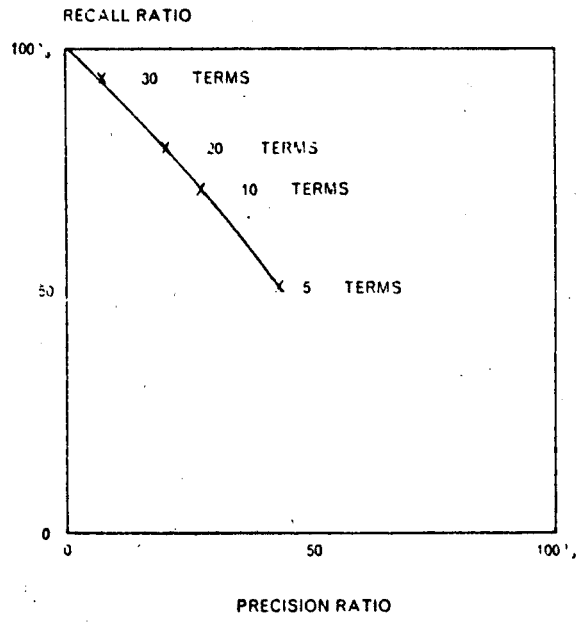
O'HARA[182] (1970) has written interestingly about the feasibility of certain regions in the precision-recall square and, in so doing, took a somewhat revolutionary step in saying that "we must assume that 0/0 = 1" which is perhaps an indication of the rigour with which mathematics has been applied to this branch of information science. In this connection, FARRADANE[183] (1974) has only recently stressed that "mathematical statisticians (must be enlisted) who are capable of elucidating valid theories on which to base evaluation procedures ... The present position is most unsatisfactory and unpromising of further advance", although AMICK[184] (1970) has applied multivariate statistical analysis to the use of a computer-aided IR system, and KOCHEN's work[105] (1974) is certainly not lacking in mathematical content. In fact, ROBERTSON[185] (1974) has taken this one stage further, by testing retrieval tests themselves (or at least applying statistical analysis). SARACEVIC[186] (1967) has isolated many variables upon which retrieval effectiveness depends, in an attempt to define a reproducible method for evaluation, and ROBERTSON[187] (1975 has intimated that all variables affecting retrieval system performance, can be analysed in terms of "M-factors" (e.g. Cleverdon's precision/recall relatio is generated by variations in a single M-factor).

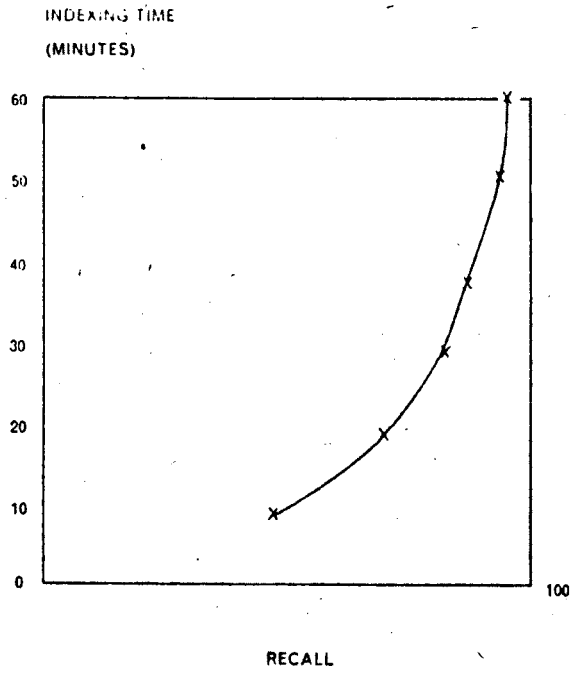
Perceptive as ever, SPARCK-JONES[188] (1975) has seen the need to assess test collections for a "good measure of performance" against which to compare systems on trial, and COOPER[189] (1973) values the user's evaluation of personal utility of a system output, as giving a "near-ideal" measure of retrieval effectiveness. The important subject of Relevance is discussed

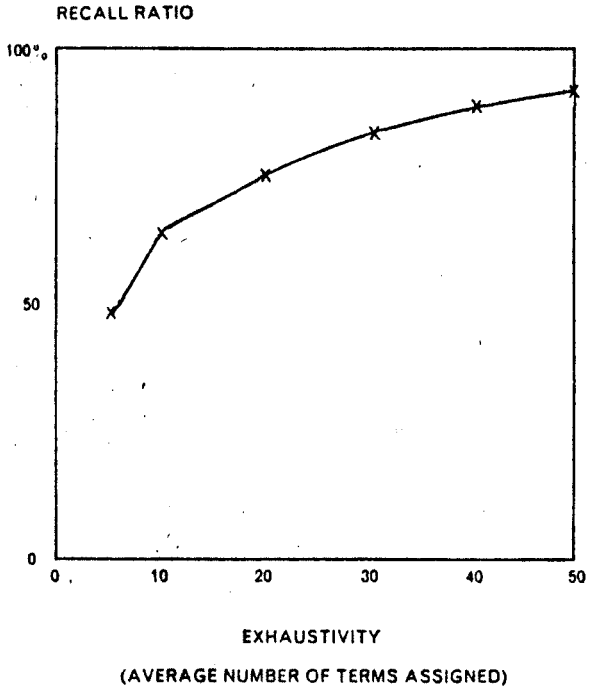
sensibly by CUADRA & KATTER[190] (1967) and exhaustively by SARACEVIC[191] (1975)

On a more practical note, SALTON[192] (1965) has tested SMART extensively, and found that high precision and recall can be obtained simultaneously, especially when a combination of methods is applied (he quotes SCHULLER[67] (1960), who found UDC and Uniterms used together to be more effective than either used alone). SALTON has also said[69] (1970), that automatic document analysis appears to be as good as that applied manually, and that user-interation ... may enable refinement of searches to given both recall and precision of 70% - rather than the present 50 - 60% - and perhaps even 80% by further sophistication of indexing and search methodology. He expands upon the importance of user-feedback in a later article (SALTON[128] (1972)), claiming 5 - 20% precision improvement at a given recall level (see fig 3.7). In this same paper he offers a well illustrated discussion on "test collection generality", and how it influences evaluation. It should be borne in mind however, that his tests were performed on the Cranfield aeronautics test collection which is, after all, confined to a pretty specific subject area.

LANCASTER[193] (1968) has considered operating efficiency vs. economic efficiency, as the basic evaluation of a retrieval system, causing the designer to face a whole series of trade-offs; for example, how good must recall and precision be if their improvement is costly? Diagrams (e.g. recall vs. indexing time) nicely illustrate this argument - fig 3.14. MILLER[194] (1971) has pointed out that it is hard to judge recall when dealing with a large collection of documents (the MEDLARS database in his case) so he has introduced on "Extension Ratio, which is approximately the ratio of the quantity of known relevant references before and after putting a query to the retrieval system". Alternatively, this Extension Ratio can be viewed as a measure of the ability of the system to multiply the quantity of relevant references known to the user.







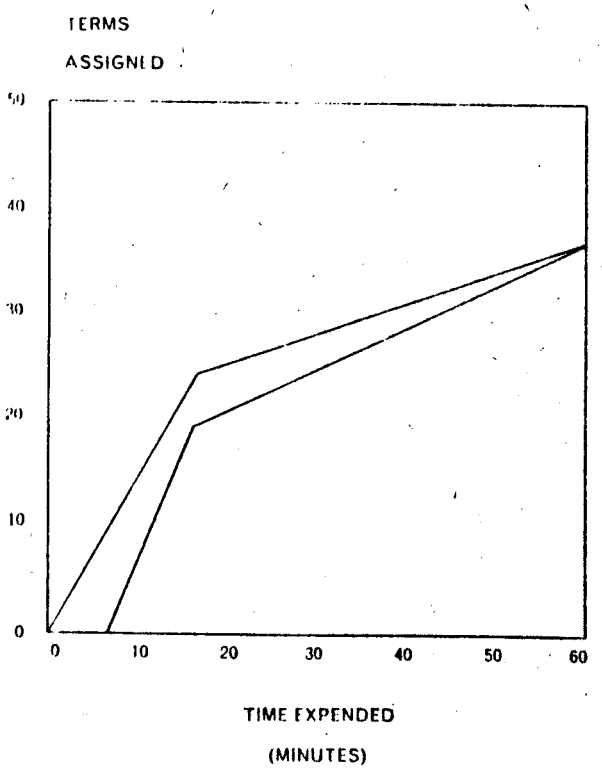


figure 3.14 - LANCASTER[193] (1968)

HERSEY[195] (1971) has made a simple, though welcome, comparison of two on-line retrieval systems, based on recall and precision measurements.

An increasingly popular (and it must be said, easier) approach to retrieval system evaluation, is that undertaken at the system level. Such evaluations and comparisons, which often involve costing, have been carried out by DAMMERS[196] (1975), ELMAN[76] (1975), LANCASTER[197] (1971), VICKERS[198] (1973), MARTYN[199] (1969), KATZER[200] (1973), STEVENS[201] (1961) and KEITH[202] (1970), whilst a SMART vs. MEDLARS comparison (auto-text processing vs. conventional indexing) carried out by SALTON[203] (1972), gave preference to SMART. A large body of opinion at the systems level favours user-oriented evaluation, which is, according to BORNSTEIN[204] (1961), free from the designers pre-conceptions, and from this standpoint KLEMPNER[205] (1964) has criticized the Cranfield project as producing only partial results since "experimental method only" was used. TAGLIACOZZO[231] (1975) has studied the utilization of MEDLINE from the consumers position, and KATZER[206] (1972) has designed adjectival scales to measure a user's reaction to the SUPARS computer retrieval system, finding it "a reliable and useful tool" for gauging user's attitudes (he doesn't reveal the terms used at the very low end of the scale).

System design theory

The theory of IR system design has a good deal in common with evaluation but suffers an even more severe lack of substantiation, to some extent brought about by a slack systems-based approach. On the more rigorous side however, COOPER[232] (1970) has attempted to formulate (and inspire further formulation of) design equations which, he feels, "when and if they became available, will be the keystones of retrieval system theory". LISTON[233] (1971) says - as do many others - that design is very complex, and should not be influenced by unfair biases and preconceptions. MARTYN & VICKERY[207] (1970) recommend an iterative approach to system design, and various other hints are contained in a review article by KATTER[179] (1969).

Summary

Having reviewed the literature, it is obvious that novel methods of Information Retrieval are emerging, but no system of so great a practical solidity that its superiority over other methods is guaranteed, has been established. UDC mechanization - mainly as examined in the AIP/UDC project - has been investigated primarily from the key handling point of view, and has thereafter been ascribed to a more lowly category of computer application than the more esoteric problems of cluster analysis.

Two areas of potential for further investigation thus seem to be of promise. Firstly, finer manipulation of UDC strings to provide a greater variety of retrieval approaches, and secondly, the pursuance of more novel techniques. Such investigations could be undertaken in the hope of proving that a combination of methods, both conventional and experimental, would be a useful contribution to an area of work in which a marked division exists between working and experimental systems, in terms of viability on the one hand, and novelty on the other.

CHAPTER 4

Problem Definition

Having expanded upon the system currently used at the Film Library, and having given some idea of the present state of the art, the simplistic "provide efficient retrieval" definition that arose in CHAPTER 1 can be modified so as to provide a realistic specification.

Retrieval requirements are best seen in the light of the types of search to which the system is subjected:-

(i) fast retrieval of a few pertinent items,

(ii) less hurried retrieval, but pre-stated requirements must be satisfied, as compared with

(iii) interactive retrieval, in which the user interacts with the catalogues to structure the eventual programmes (ie no hard and fast retrieval demands exist prior to searching, and the process is generally guided by feedback.)

In terms of physical retrieval demands, the above requirements can be translated into terms having direct impact upon search strategy:-

(1) fast first-access to approximate target area,

(2) expansion around first-access,

(3) alternative search methods (if (i) & (ii) fail), and

(4) unconstrained meandering about the collection.

The most common type of search at the Film Library (and that which all retrievals involve at some stage or other) is the fast first-access retrieval that puts the librarian in the target area - hence the use of general UDC strings as highlighted in CHAPTER 2, fig. 2.17. Note

that this practice runs contrary to that advised during training, when librarians are advised to adopt high specificity from the outset). This is followed by browsing in order to refine the low relevance, high recall stock that usually results from a general first search. (A specific search would increase relevance, but perhaps depress recall to such an extent that enquiry reformulation would be necessary, hence the unpopularity of high specificity.

Given that UDC indexing at the general level is easier, as shown in Chapter 2, a broad UDC search is likely to constitute an efficient means of gaining access to the collection, and since the UDC is structured (in that similar subjects are filed together), expansion upon the original search is easy. Also, since the UDC is hierarchical, more powerful possibilities exist by way of broadening or narrowing original queries. Despite possible structural faults therefore, the UDC does offer the possibility of fast first-access, to be followed either by UDC-based search refinement or, given a manageable number of cards, a visual scan of the individual prose entries (also known as "analytical entries") which I believe to be one of the most efficient retrieval mechanisms that I've encountered.

When one talks of "alternative search methods" at the Film Library, one invariably refers to query type-switching manoevres (Subject to Name, Subject to Title etc.), whereby the librarian's skill lies in reformulating the original query so that it might be approached by another access method. For example, given a failed search on "anti-gravity devices", one might try scanning through the "Tomorrows World" entries by way of a Subject to Title switch. Alternative access to the subject data is only possible by query reformulation.

It can be seen therefore, that the UDC is well suited to current retrieval requirements, but one can never be sure to what extent "requirements" are constrained by availability of alternatives.

Subject Classification

From the previous chapter, one can see that four distinct methods of subject classification exist:-

(i) the classical methods, eg UDC, Dewey etc.

(ii) methods requiring intellectual involvement, but heavily geared towards computerization, eg. PRECIS

(iii) Keyword classifications - wholly mechanical, requiring a minimum of intellect and relying upon a stable term vocabulary.

(iv) clustering systems etc. - wholly computerized, using sophisticated methods to reduce rigidity of the vocabulary, and hence to encompass a natural language orientation.

Of (i) to (iv) above, the greatest promise is to be found in the ambitious cluster type systems that are constantly being improved. The current faults of such systems mostly be in their inefficiency, and when compared with computerization of an established system such as the UDC, one is soon convinced that the more intellectual indexing effort one is prepared to put in, the less is the demand put on the computer.

As an aside, it might be mentioned that I have been told (by people at the Film Library) that a media library, more than any other, positively demands an intellectually based subject classification, in order to make up for the degradation that is implicit in storing

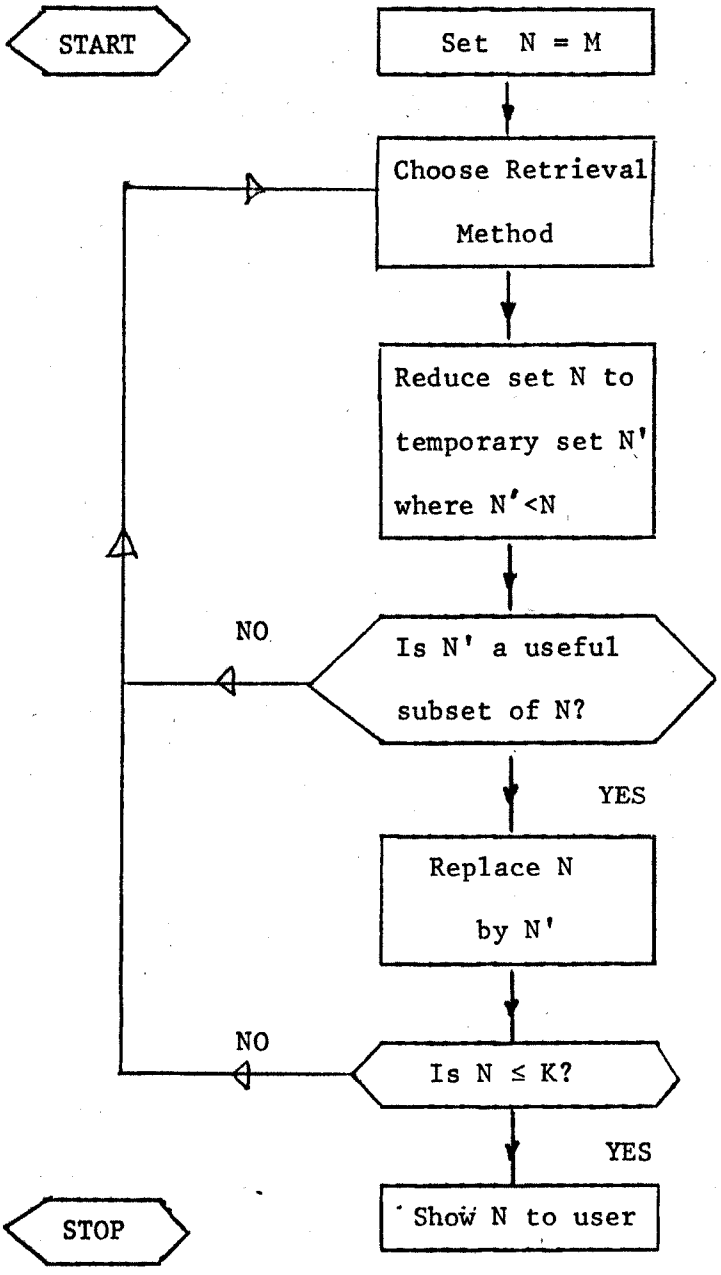
non-word material via written descriptions. This isn't a point that one can either definitively prove or disprove, but as long as it is believed by those responsible for running the Film Library, I think it is an important point.

Any reasonable appraisal of IR methods eventually leads to the conclusion that all established methods have comparative (and even absolute) advantages and disadvantages. What is more, under certain conditions, unconventional methods can be distinctly superior to those that are established. To pick a single storage and retrieval system from a list of likely candidates, is perhaps the most singly destructive phase in the design of any such system.

Composite Retrieval Systems

Given retrieval systems A,B and C , in a particular situation one of these will undoubtedly be of superior value, according to the user's criteria of necessity. That is to say, user X might pick system A for the sake of speed, whereas user Y will pick system C for the sake of comprehensive coverage. In different situations and with different users (or query types - see earlier) A,B and C will shift in terms of their relative merits, and if the user were to be given recourse to a choice of retrieval methods during a single retrieval search, the benefits of using a composite approach could be expected to be great.

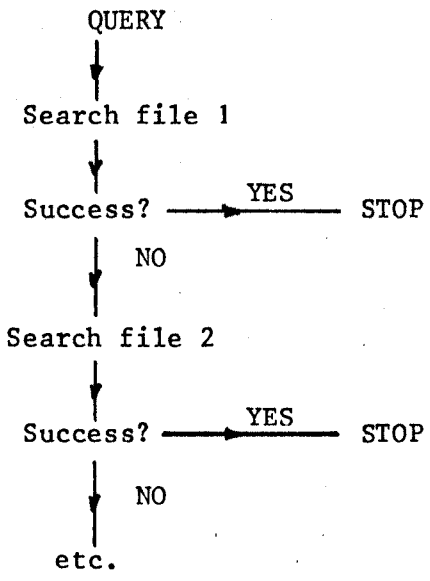
In this respect, it is useful to regard IR as a sequence of partitioning operations, starting with a set size M (= the total collection) and concluding with a set size K, which is of such size that one can browse through it manually to make a final selection. A retrieval step is any device by which a set N might be reduced in size, and retrieval is complete when $N \leq K$. Information retrieval thus proceeds as follows:-



In CHAPTERS 5,6 and 7, various approaches to retrieval will be discussed that, in essence, constitute prospective components of such a modular system, and in CHAPTER 8 a more comprehensive discussion of this approach will be undertaken. However, there is an important point which might usefully be broached now, since it concerns the format of a total retrieval package as it might be implemented at the Film Library.

Various retrieval paths, corresponding to the various enquiry types, have already been mentioned (Subject, Title and Name), as has the purposeful switching of enquiry types to recover failures, and allusion

has also been made to other sources of information at the Film Library. In short, the Film Library consists of a wide range of information files which can be selectively examined in the process of retrieving a piece of film from the vaults. These files, generally considered as discrete entities (to suppose otherwise would be to imply redundancy) of course share one common attribute, in that all are centred around the aim of film item retrieval, but they are seldom likely to be used concertively in achieving this goal, since manual comparison of large files is not a trivial matter. In fact, more than one information source will only usually be searched if the methods chosen in preference have proved inadequate, ie:-



This approach manifests two major disadvantages:-

(i) it is lengthy, and

(ii) unless all the relevant information sources are consulted, something less than the "best" retrieval is likely to be achieved.

A computer system however, might be designed to make concertive use of files possible, as an addition to the parallel approach of the modular retrieval method. In other words, I seem to be promulgating the design of a system in which a range of files and retrieval devices are thrown

together like building bricks, and used for retrieval as best suited to the individual enquiry. Such an idea simply begs consideration of the Relational Approach to data storage and manipulation (see CHAPTER 8).

For the moment however, the importance of the preceding paragraphs is that they encourage investigation of diverse retrieval techniques which can be brought together into one consolidated system, wherein individual defects are reduced. The next three chapters are used to describe three specific retrieval methods (building blocks) upon which such a modular construction could be based. In CHAPTER 5, a computerized UDC system, having the fast access advantage discussed above, is detailed. The methods described in CHAPTERS 6 & 7 complement the UDC programs by way of alternative strategies, so that in a combined system, individual weak points can be by-passed, and the full range of retrieval requirements met by controlled structuring of the system components.

CHAPTER 5

The design and construction of an on-line, UDC-based pilot information storage and retrieval system

This Chapter contains a description of the design, production and testing of a wholly UDC-based system, and, as such, one that closely mirrored the existing manual methods used at the Film Library. In succeeding chapters, other ideas for retrieval will be described which have little or no connection with the UDC, and, it is hoped, embrace a higher degree of novelty and utility. For the moment however, let me list the reasons by which the design of a purely UDC based system is justified:-

(1) Being based on an existing manual system, a computerized version would:-

(i) calm Film Library personnel (who, as is to be expected, question the reliability of any system, especially a computer system, with which they are not conversant), and assure them that they would be able to use the new system (in trials) with the minimum degree of training;

(ii) be capable of a level of success at least as great as that achieved by the manual system (given a straightforward copying of facilities); and

(iii) permit more complex search strategies to be used, that were simply out of the question when operating the UDC manually.

(2) As an extension of 1(iii) above, there is always the chance that computerization of a manual process will lead to unforseen possibilities (see CHAPTER 6 on the LEARNER program, which grew out of the pilot system).

(3) By the use of recent software developments, it was expected that a tailor-made retrieval system could achieve a high level of efficiency, with a greater diversity of operation than that to be found in earlier systems (eg AUDACIOUS [224]).

(4) A UDC system could be viewed as a possible component of a more comprehensive modular retrieval approach (see CHAPTERS 4 & 8)

(5) Design of a UDC system proved a good starting point for what was essentially a practical project.

In fact looking back, building this UDC based system was such an obvious phase of the project, that to have omitted it would have been unthinkable. On a practical note, no user could be expected to unreservedly welcome a solution that had ignored the major alternative offered by their existing manual system, and so "defence of credibility" should be added to the five points listed above.

It was intended that a pilot system should evolve so that:

(i) the adequacy of UDC numbers as retrieval keys reflecting their conceptual content could be proven, and that

(ii) Film Library personnel should be able to test the system, firstly to help in its design, and secondly to assist in their familiarization with computer methods.

As soon as the software was written, a terminal was installed at the Film Library and the computerized system was introduced, selectively at first, but then to a large number of the librarians who would be the potential users of a production system.

The aim was to produce a system for information storage and retrieval based on UDC keys, and so only the three files central to the UDC organization needed to be considered, namely:

(i) the main UDC catalogue itself

(ii) the Subject Index (for Subject-UDC translation), and

(iii) the Authority File (for UDC-Subject translation)

The end point in the Film Library subject retrieval process is a set of UDC catalogue cards which refer directly or indirectly to film reels. These cards contain a UDC number and a piece of descriptive prose (see Fig. 1.5a) which may be over a hundred words long, and although librarians claim that the UDC code itself is rich in meaningful mnemonics, there is little doubt that it is the prose which determines selection or rejection of the retrieved items. However, to include this prose in a pilot study would either have involved a greater data conversion cost than anticipated at this early stage, or a forced reduction in the number of individual items to be incorporated in the pilot. It was decided therefore, to omit the prose altogether, which although perhaps diminishing the reality of the pilot, in no way interfered with the UDC retrieval aspect that was under test. What is

more, I was assured by Film Library staff that a UDC code presented as the result of a retrieval search would be almost as meaningful (to them) as a piece of prose. Of course, in any production system the prose would be a vital component, and could be incorporated in the present pilot system in a trivial step (Note - the UDC apart, the card prose is a separate source of information - see CHAPTER 7).

The three files to be computerized were therefore as follows:-

UDC catalogue - filed in UDC order

Field 1 UDC Code	Field 2 Card Accession No.
34	2986
34	4369
341	2172

Subject Index - filed alphabetically on subject

1 Subject	2 UDC
Int. Law	341
Law	34
Terrorism	343.77

Authority File - filed in UDC order

1 UDC	2 Subject
34	Law
341	Int.Law
343.77	Terrorism

For the sake of the pilot study, the following areas were put on the computer:-

UDC area	Number of UDC strings
34 to 341.358.007	3348
[AEI] to [LADBROOKES]	361
(480) to (491.1 WESTMAN)	403
(=20) to (=943.5)	453
M74:623.451 to M99 "414.22"	460
R13(256) to R13(421)	339
	Total = 5564

Data Structure

Before embarking upon the design of a retrieval system, one is well advised to take a long, hard look at databases. Buying in a manufacturer's database can offer a number of distinct advantages:-

(i) established and well supported software,

(ii) standardized manipulation procedures, and hence "programmer hospitability" as it is known,

(iii) assured security and error recovery procedures,

(iv) and perhaps even a conceptually sound framework

Individual systems will cover these and other advantages to greater or lesser degrees.

At the present time, databases fall into two broad types. Firstly there are the commonly used hierarchical structures, in which relationships between data items are pre-defined. Databases of this type are now to be found all over the place, and are those most commonly provided

by computer manufacturers. Standards of structure and operation are guided by CODASYL edicts. Secondly there are Relational databases of which I will write more (see Chapter 8). Compared to their hierarchical brethren, production versions of relational databases are rare, and their flexibility makes them less efficient than their counterparts. In terms of standardization however, obeyance of the precepts underlying the relational approach ensures that any truly relational database will unavoidably have a great deal in common with any other such database, and so impositions by an arbitrary standardizing body would seem unnecessary.

In fact, a tailor-made retrieval and updating system was built in preference to the straightforward adoption of one of these database approaches. As it happens, this step in no way hinders the potential for simultaneous application of a relational system at the Film Library, and this interesting dichotomy will be fully discussed in CHAPTER 8. The hierarchical approach was, however, rejected for the following reasons:-

(1) UDC retrieval, which is the prime area for computerization at the Film Library, poses so specific a problem that a tailor-made system offers advantages of:-

  (i) lower software overheads

  (ii) greater concentration of purpose and hence greater efficiency.

(2) The wide diversity of application to be found in hierarchical databases at the cost of computer resources, would not be needed at the Film Library.

(3) The Employee-Name/Employee-Number type relationships around which hierarchical databases were designed to operate, have little in common with the complex tree structure to be found in the UDC (which would be most difficult to express in CODASYL terms).

(4) Purely from the practical point of view, the pilot study was to be run on a UNIVAC machine, whereas any production system would be ICL based. In the hope of maintaining transportability of software, reliance upon common standards of database manipulation was not thought to be desirable.

* * * * *

In going for a tailor-made system, it was therefore necessary to decide upon the best manner of structuring the three files for storage and retrieval, so that the following could be achieved:-

(i) rapid retrieval via UDC number (UDC catalogue and Authority File) and via literal string (Subject Index),

(ii) efficient updating, preferably on-line,

and that this should be possible:-

(iii) by using algorithms that could be efficiently expressed in COBOL (the language favoured by the BBC).

Before going into details of file structure however, it is necessary to consider the nature of a UDC number, and the manner in which its integrity must be preserved in becoming a retrieval key.

Unadulterated, the UDC filing order does not match any recognized computer scheme for string evaluation (ASCII in the case of Univac COBOL), so it was necessary to produce a subprogram capable of recoding UDC strings in such a way that both the UDC schedule and the ASCII conventions were satisfied. It was also necessary however, to preserve the meaning of the UDC strings as conveyed by their mnemonics and, most important of all, to maintain the detectability of facets.

Suffice it to say that this was achieved, although it took some time to cater for all the vagaries of the UDC filing order. Any UDC number being handled by the system was first therefore translated by the computer, so as to satisfy its own filing conventions, but without in any way reducing the information content of the string.

Given that this was possible, a UDC key could then be treated quite as straightforwardly as any piece of literal data, but with the additional advantage that a recoded UDC string could be interrogated with respect to its symbolic content, and hence its physical meaning.

File Structure

Anyone confronted with the design of a file structure from scratch, especially when new to it, is faced with a giddying array of alternatives. First there are the simple approaches (linear files, binary files) that promise ready implementation but numbing inflexibility. Then come the more versatile systems (inverted files, chained lists) that are often to be found in "buyable" packages, but offer only selected advantages over their more primitive rivals. Next come the proven stalwarts, such as hash coding, that have the potential for greater efficiency, but only when they are amenable to the problem under examination. Lastly come the true data structures (trees, networks) which, if correctly chosen, give immense flexibility and efficiency, but suffer the need for complicated software.

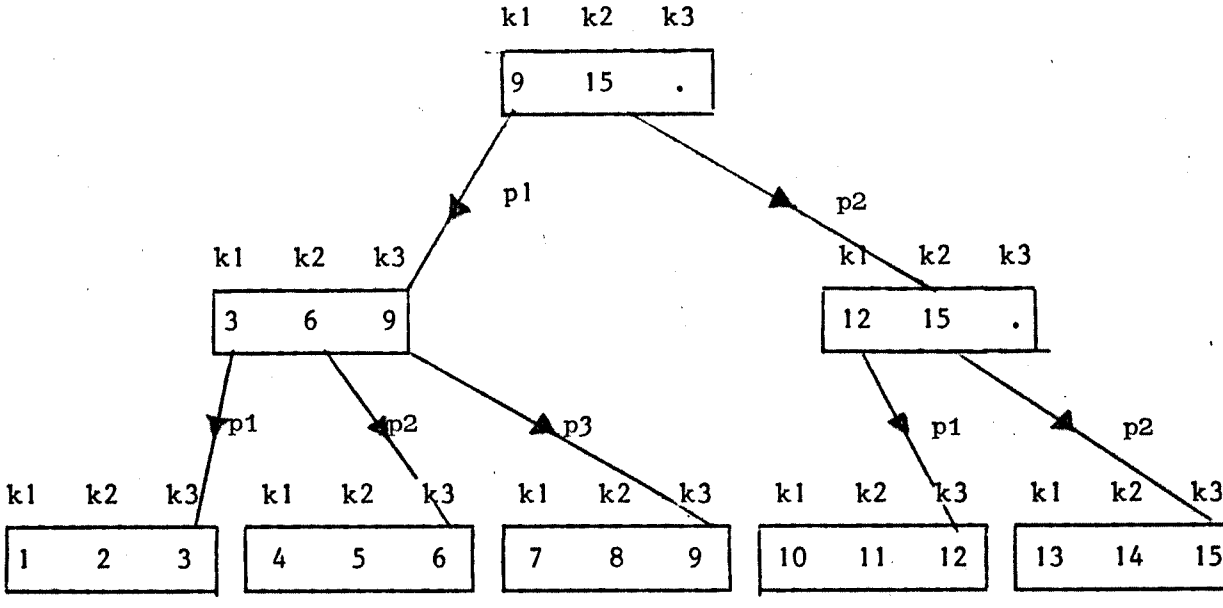
When designing a specific system, one has no excuse for picking an easy alternative unless it genuinely comprises the best option, and this is seldom the case. Certainly, given my problem, the size of the data collection (close on half a million items) and the need for rapid retrieval and updating, caused rejection of anything in the first two categories above. Also, hash-coding, a good choice in the right circumstances, seemed particularly unsuited to handling UDC strings, the

idiosyncracies of which would have necessitated a most complex algorithm, and one unlikely to achieve good space management.

In short, the last alternative seemed, from the outset, the most attractive, with trees, and in particular B*-trees, emerging as the most likely candidate. In choosing a hierarchical structure, there was no attempt to mirror the hierarchy to be found in the UDC - the differences between a logical tree and a conceptual UDC hierarchy being two profound to plumb. A tree structure was selected purely on grounds of efficiency - a quality which is pre-eminent in the B*-tree, and is supported by other characteristics of this structure that particularly suit the Film Library files.

The theory behind B-trees and predictions of their performance is contained in the original paper by BAYER and McCREIGHT [238]. This is further expanded upon by others [240 to 243] and notably by KNUTH [239], who discusses the particular variant of the B-tree used in the Film Library programs - christened by KNUTH the "B* tree."

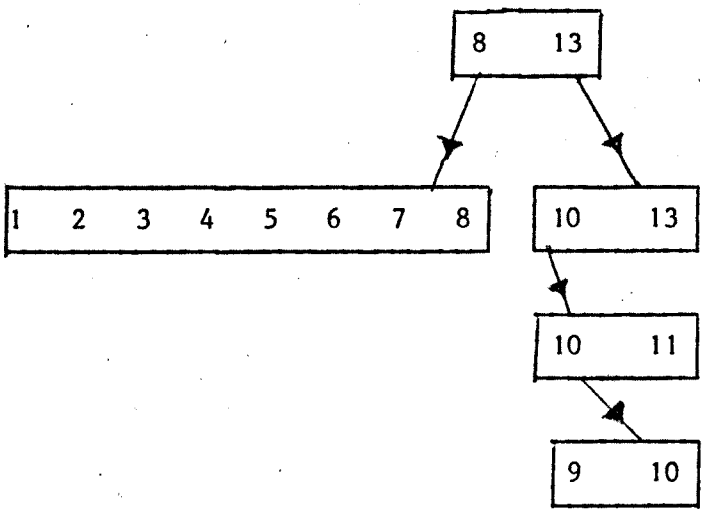
Consider a paged file in which each page contains a maximum of 3 key/pointer pairs (ki/pi). The numbers 1 to 15 could be held as a B* tree in such a file as follows:-



The pointers on the leaf pages (those at the lowest level) can point to another file where other information can be stored (text etc.). Retrieving information associated with any of the above numbers would require 3 page accesses to be made in the tree.

Each page in a B^*-tree contains a maximum of "N" keys, so when the tree is full, "p" page accesses are required to locate 1 in N^p items. Of course, the tree will seldom (if ever) be full, but the theory associated with the B-tree structure enables prediction of tree utilization. In fact, it is the strategy with which the tree is constructed and maintained that dictates the efficacy of the structure.

The "B" in B-tree stands for "balanced". An un-balanced tree would look like this:



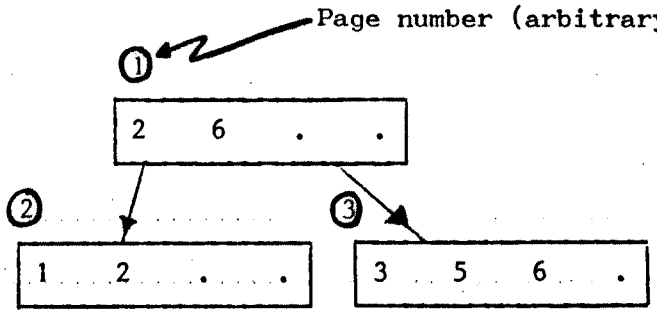
Retrieval of "4" takes 2 page accesses

Retrieval of "9" takes 4 page accesses

The B-tree maintains its balance however, by shifting keys around at a level, rather than deepening or reducing a hierarchy. For this purpose, each page contains pointers to its right and left brothers. In the following examples, the minimum page size = 2 keys and the maximum = 4 keys:-

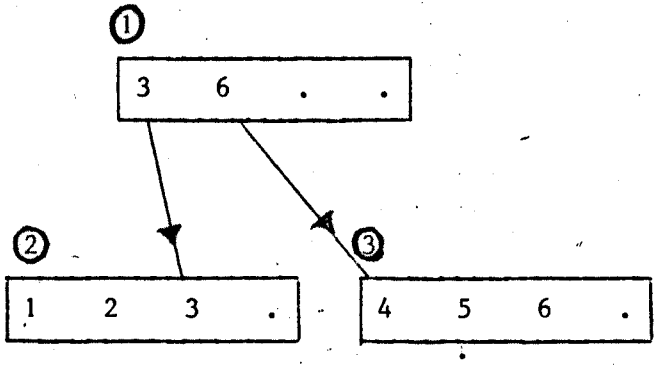
Insertion

(1) BEFORE:



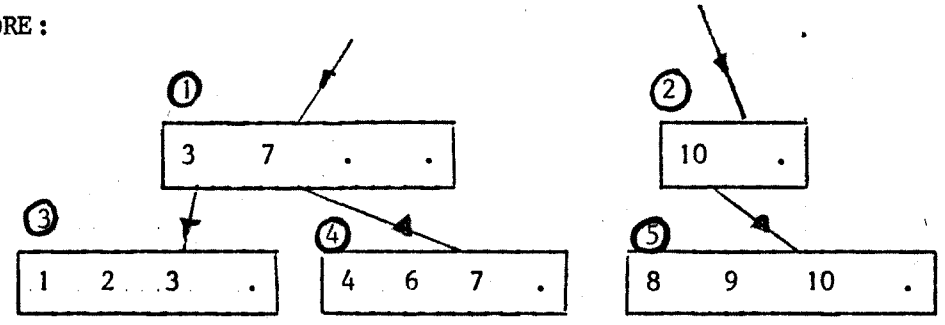
UPDATE : Insert key "4"

AFTER:



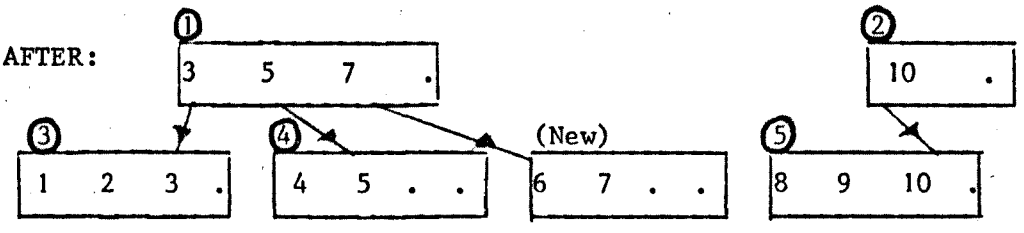
ACTION: Adding "4" gave max. keys on page ③, so ③ was over-flowed into brother ② and the father ① was updated accordingly.

(2) BEFORE:



UPDATE : Insert key "5"

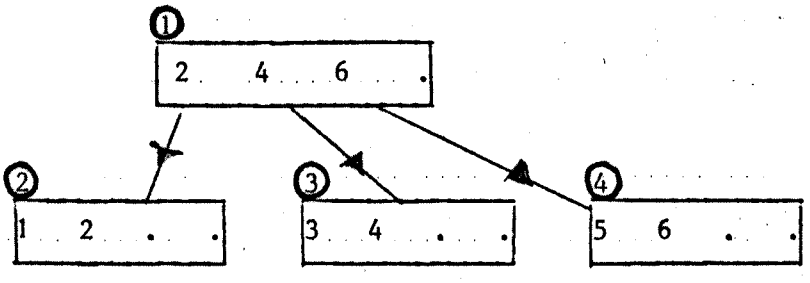
AFTER:



ACTION: Adding "5" gives max. keys on ④ . Page ④ can't be overflowed to left ③ or right ④ without causing a full-page condition, so page ④ is "split" (causing a new page to be formed), and the father page ① is updated.

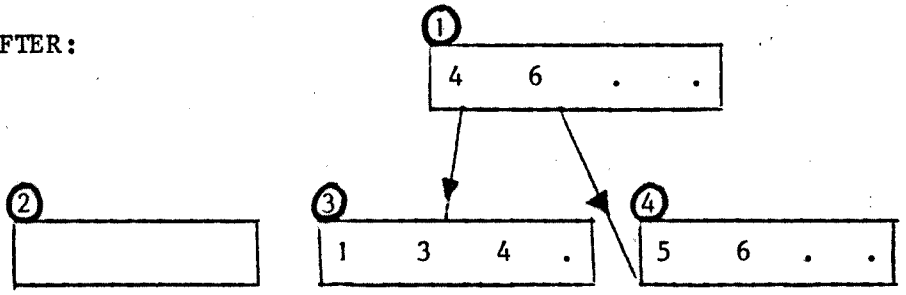
Deletion

(1) BEFORE:



UPDATE: Delete key "2"

AFTER:



ACTION: Deleting "2" means that pages ② and ③ can be combined on ③ . Page ② is made available for re-use and the father page ① is updated.

Updating always begins at leaf level, but is might be necessary to work back all the way up the tree to correct successive parent pages, even as far as the uppermost or "root" page.

In the program for maintaining the Film Library files ("UPDATE"), page accession is under the control of a routine that permits a maximum of five pages to be held in core at any one time. Therefore page I/O is virtual, in that a READ PAGE command does not necessarily result in a physical READ, since the page may already be in core. Pages are given a "priority value" such that the root page has the highest priority, and the leaf pages the lowest. Pages with the higher priority values are, on average, required more often than pages with lower values (the root page is required for every search), so the justification for a page to be kept in core is proportional to its priority value, and pages with the lowest value are the first to be overwritten (in core) once they have been used. The program for retrieving from the Film Library files ("RETRIEVE") only allows one page to be stored in core at a time, since retrieval proceeds page by page, and merging etc. is not necessary.

The capacity of a B* tree under varying conditions of tree height and page size is given in figure 5.1 below:-

Maximum Number of Keys per Page	Tree Height = 1		Tree Height = 3		Tree Height = 5	
	Minimum Capacity	Maximum Capacity	Min.	Max.	Min.	Max.
30	30	30	6.7×10^3	2.7×10^4	1.5×10^6	2.4×10^7
60	60	60	5.4×10^4	2.2×10^5	4.9×10^7	7.8×10^8
100	100	100	2.5×10^5	10^6	6.3×10^8	10^{10}

figure 5.1

Figure 5.1(a) below is reproduced from the original paper by BAYER & McCREIGHT[238], and contains predictions of B-tree performance in respect of retrieval, insertion and deletion:-

Organization and Maintenance of Large Ordered Indexes

	Retrieval	Insertion in index without deletions and without overflows	Deletion in index without insertions, with or without overflows	Insertion in index without deletions, but with overflow	Insertion in index with deletions, without overflow	Deletion in index with insertions, with or without overflows	Insertion in index with deletion, with overflow
min	$f = 1$ $w = 0$	$f = h$ $w = 1$	$f = h$ $w = 1$	$f = h$ $w = 1$	$f = h$ $w = 1$	$f = h$ $w = 1$	$f = h$ $w = 1$
Average as derived in paper	$f \leqq h$ $w = 0$	$f = h$ $w < 1 + \frac{2}{k}$	$f < h + 1 + \frac{1}{k}$ $w < 4 + \frac{2}{k}$	$f \leqq h + 2 + \frac{2}{k}$ $w \leqq 3 + \frac{2}{k}$	$f = h$ $w \leqq 2h + 1$	$f \leqq 2h - 1$ $h - 1 \leqq u \leqq h + 1$	$f \leqq 3h - 2$ $w \leqq 2h + 1$
max	$f = h$ $w = 0$	$f = h$ $w = 2h + 1$	$f = 2h - 1$ $w = h + 1$	$f = 3h - 2$ $w = 2h + 1$	$f = h$ $w = 2h + 1$	$f = 2h - 1$ $w = h + 1$	$f = 3h - 2$ $w = 2h + 1$

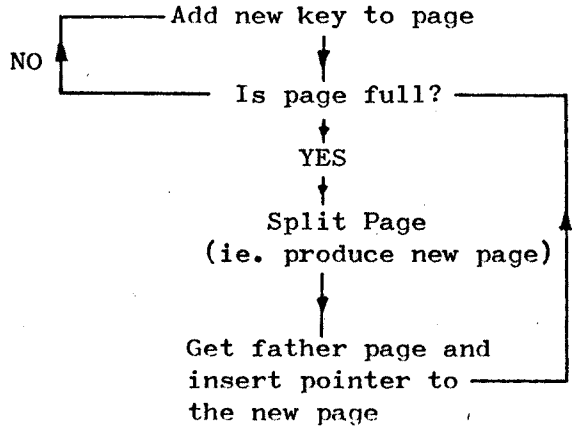
f = number of pages fetched — h = height of B-tree
w = number of pages written — k = parameter of B-tree of pages
I = size of index set — u = best upper bound obtainable for w

Table of costs for a single retrieval, insertion, or deletion of a key

figure 5.1(a)

WEDEKIND[23] has assessed B* trees as being superior to B-trees performance-wise, in that average tree height is reduced ("h" in the previous table).

When constructing the Film Library B* trees, the data was already sorted, and so to have built the trees from scratch by means of the standard insertion algorithm would have been extremely wasteful. In fact, B-tree insertion of pre-sorted data would have proceeded as follows:-



which would result in a sequence of half-filled leaf pages. To achieve the same result from pre-sorted data, it was possible to first build the leaf pages in sequence, followed by the higher levels, simply by writing out a page when it became half full and then going on to the next, thus enabling a great saving in resources to be made.

The decision to produce a sparse tree, rather than one with say 66% or even 100% storage utilization, was taken in the hope of providing plenty of space for insertion without splitting during the young life of the tree, since insertion is by far the dominant process in the Film Library files. With a 50% utilization, a tree was built of height 3 (at the maximum capacity of 30 keys per page in all of the B* trees, the parameter "k" in fig 5.1a is 15), whereas a full tree at the leaves would need to be of the same height to hold five and a half thousand keys, but would be that much less demanding in terms of disc storage space. The full tree would, however, be prone to a maximum degree of page splitting, whilst

figures for the sparse tree would be at worst average and, to begin with at least, minimal. Substituting in the fourth column of figure 5.1(a) therefore, one can see that for the sparse tree, values such as:

$$f \leq 3 + 2 + 2/15 \quad \text{(no. of pages fetched)}$$
$$\text{and} \quad w \leq 3 + 2/15 \quad \text{(no. of pages written)}$$

would obtain (even as a pessimistic estimate), whereas for the full tree, values such as:

$$f = 9 - 2$$
$$\text{and} \quad w = 6 + 1$$

would be more likely, thus giving a distinct edge to insertion in a sparse tree. In time of course, a sparse tree or a full tree would tend towards some average storage utilization by virtue of the insertion strategy (page overflow), but initial conditions can be favourably adjusted when handling pre-sorted data so as to anticipate likely system use, assuming that a certain lack of thrift in disc management can be tolerated.

In summary, it can be said that the B^*-tree provides a balanced structure in which retrieval speeds can be predicted, and which updating is localized. Particularly suited to the Film Library - where thousands of keys, many of them duplicates, need to be rapidly scanned in sequence (eg. in QSEARCHing - see later) - is the characteristic of the B^*tree (though not of the B-tree), wherein a sequential presentation of the keys on the leaf pages is maintained by means of brother-to-brother pointers.

Building the system

Before the B^* trees were implemented in the three files being computerized, a large amount of system design was performed using binary files, in order to define the general retrieval requirements (search facilities,

retrieval language etc.) As has been said, design was based firmly on the existing manual system, and so the schematic outline description in figure 5.2 would not be unfamiliar to a librarian.

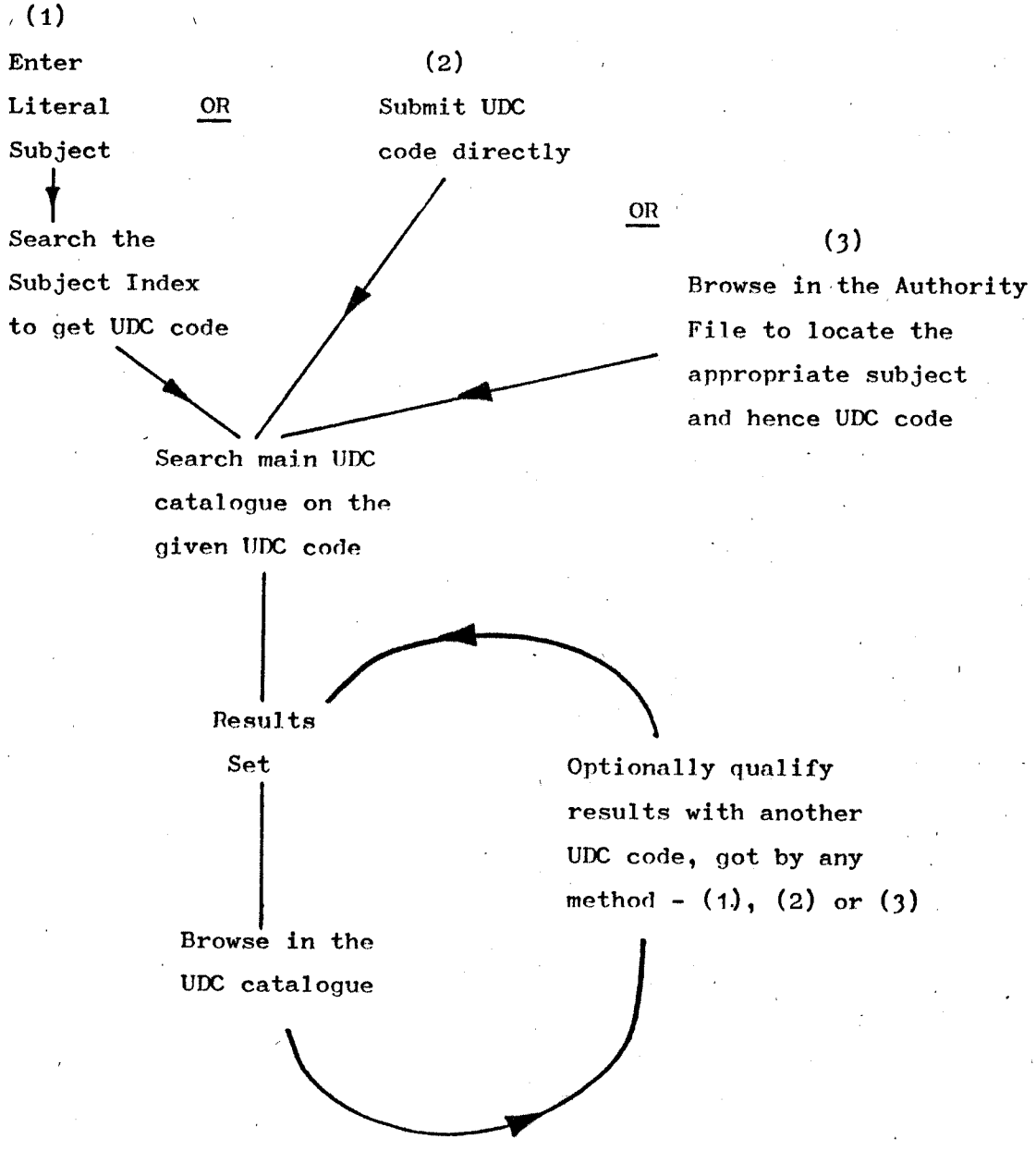
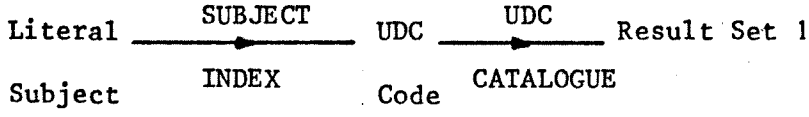


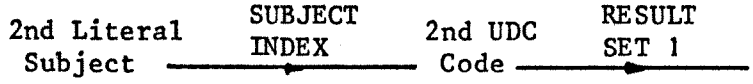
figure 5.2 - Outline computerized retrieval process

No profound research was applied to the formulation of a command language, and eventually a simple two level system was developed using short commands of minimum technicality. Details of the programs will be given shortly (the user manuals are included as APPENDICES 1 and 2) but first I will describe the general properties and capabilities of the retrieval system, and the manner in which design was influenced by reaction from the Film Library.

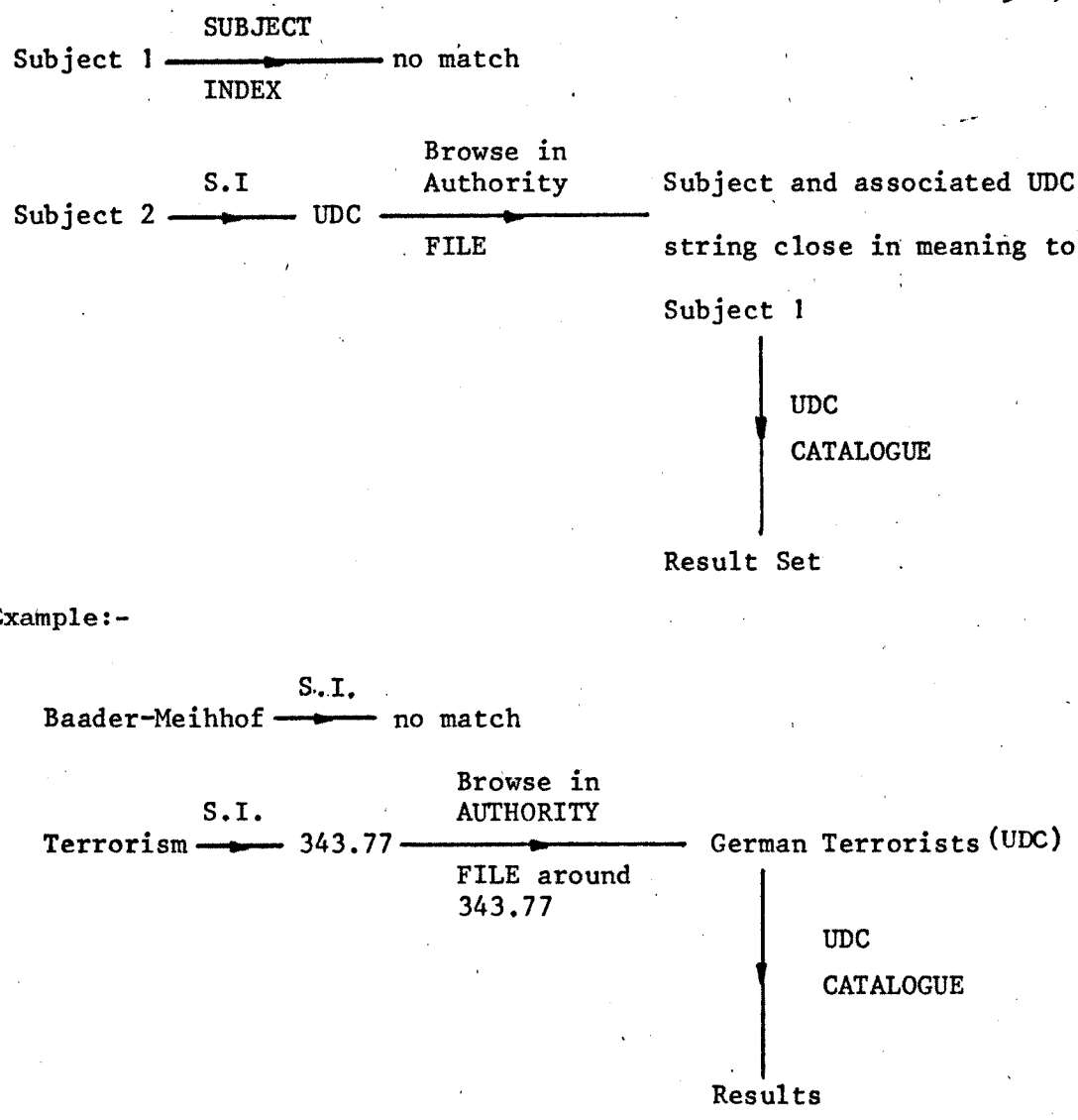
The three files were loaded on to the computer as B*-trees with fixed length keys. (A recent paper concerning variable length records in B* trees could usefully influence future system design changes - see McCreight [242]). Additional files were necessary for the storage of auxiliary information, but searches were confined solely to the fast access B*-trees. It was anticipated (from the statistical survey and my own observations) that retrieval would most commonly take the following route:-



The Result Set, if of excessive size, would then be cut down by qualification on another UDC code, thus:-



If the Subject Index had proven inadequate as a purely alphabetically organized subject tabulation, a user would commonly substitute a synonym for the original literal subject and try the whole process again. Failing this, recourse would be made to the file organized by subject, namely the Authority File, in hope of procur ing a synonymous subject. First access to the Authority File would commonly be by way of a term (Subject 2) less specific than the original term (Subject 1), ie:-



Example:-

Whatever the method, the first retrieval method would inevitably be the divination of the relevant UDC code, and the computer system provides for this by means of either the TERM, MYUDC or AFILE command (options 1, 2 or 3 in fig 5.2). The use of these three alternative subject selection commands is as follows:-

for TERM :- (i) user enters literal subject

(ii) computer searches Subject Index B*-tree for nearest match

(iii) user can browse in Subject Index

for MYUDC :- (i) user enters a UDC number direct (librarians tend to remember the more frequently occurring strings).

for AFILE :- (i) at some point, user switches from an entry in the Subject Index to the equivalent entry in the Authority File

(ii) user can browse in the Authority File to find the concept closest to the original query.

Since the Authority File is organized by subject, the user is able to move up or down the UDC hierarchy by choosing options attached to the AFILE command.

So, having found a suitable UDC code, the user then goes on to the:-

SEARCH command:- (i) computer searches UDC B* tree for matches to the selected UDC key

(ii) user is informed of the number of matches obtained.

Matches between UDC file entries and the search key are defined as follows:-

(1) If search key = file key - a direct hit is registered

eg 341.123 341.123
Search File

(2) If search key = file key, but the file key then goes on to be more specific - a root hit is registered

eg. 34 34:621.2 (ie. 34 in common)
Search File

Having SEARCHed, the user can then go on to browse in the relevant area of the UDC file by means of the:

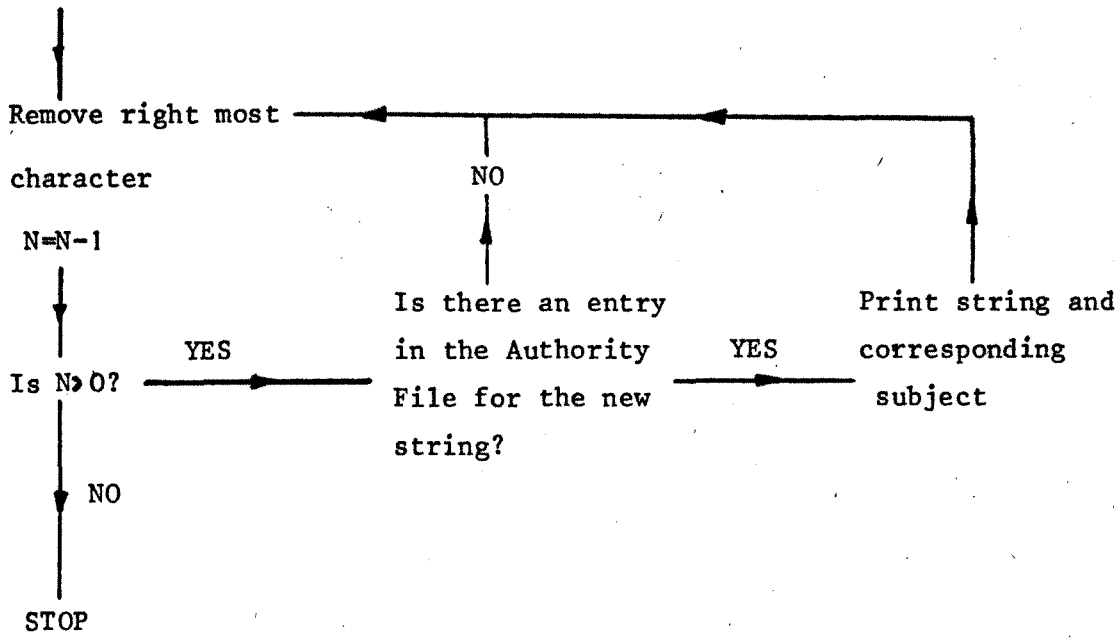
BROWSE command:- (i) computer positions user in the results set (direct and root hits)

(ii) user can browse within and outside the results set, ranging over any area of the UDC file (using low level "browse" commands)

As BROWSING proceeds, all UDC file numbers encountered are matched against the original search key and a measure of similarity computed, which is printed out for the user's reference, eg:-

Search Key	UDC Key	Similarity Score %	
341	341	100	Items within results
341	341.123	100	set
341	34.096	66	
341	3	33	

The steps outlined above comprise the basic search functions built into the pilot system. The computer records details of all UDC search strings and search operations themselves, so that the user has a ready reference to what they've already done. The hierarchical browsing that I alluded to in the case of the Authority File does not imply that the UDC structure is explicitly stored. In fact the manner of this browsing serves to flatter the high retrieval efficiency of the B* tree. For example, to move up the subject hierarchy from UDC string 341.123, the program knocks off digits from the right and searches for a match to the newly created string (341.12, 341.1 etc.), printing it if it corresponds to an existing subject entry. This algorithmn can be expressed as follows:-



Working down the hierarchy is more complicated, because it involves synthesizing strings by the addition of characters to form new, legal UDC numbers. This lengthy process is made viable thanks to the rapid retrieval speeds associated with the Authority File B* tree. Starting with UDC = 3 (Social Science) for example, all possible extensions can be constructed and sought for in the Authority File (31,32,33 etc.), so descending the hierarchy a level at a time. The user can then choose any of the valid routes traced by the computer, and thus progress to the immediately inferior sequence of nodes, repeating the process until all existing downward paths are exhausted. (Hierarchical browsing in the Authority file is discussed in detail in the RETRIEVE program user's guide - see APPENDIX 1).

In whatever manner the first retrieval is performed, there is always the chance that it will not have been a total success. If an inadequate results set is derived, the librarian commonly reformulates the query and tries another approach, since the computer can do little to compensate for subjects about which it has little or no data. If however, a search produces an unmanageably large results set (one that

cannot be browsed through conveniently) the computer can be expected to help, and help it does by means of the:-

QSEARCH command ("qualifying search"):-

(i) user supplies a second UDC string (by any of the methods outlined earlier, eg. TERM, MYUDC etc.)

(ii) computer searches the most recent results set for keys that also contain this new UDC number

(iii) a more refined results set is formed and displayed.

QSEARCH involves string comparison, by which the qualifying string is checked for occurrence within keys held in the results set being QSEARCHed. Any UDC string can be used for a QSEARCH, but only "legal" occurrences are registered. For example, to QSEARCH by 67 on string 343:67 (42) would register a success, whereas string 367.29 should not and would not satisfy the QSEARCH, since a substring embedded within a facet has no conceptual significance,

eg. in 342.978 and 343.77 : 62.78

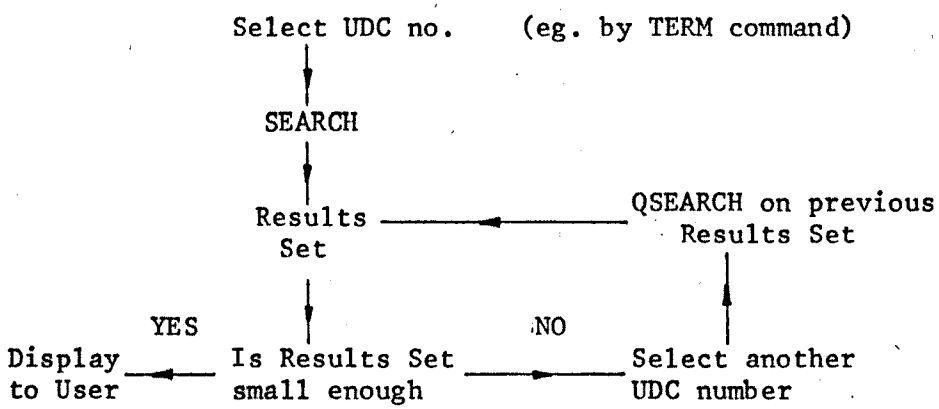
the 34 group has a common significance, but the 78 group means nothing unless the context is taken into account.

In other words, a QSEARCH would most commonly be used to interrogate a results set for the existence of facets within it that were not the subject of a previous SEARCH. However, certain auxiliary groups, which are not themselves facets, convey a meaning that is independent of position (the same can be said of facets) so QSEARCHing on these auxiliaries - eg..007 for people - must be legal.

QSEARCHing therefore required the production of a string comparison subprogram that took account of all these considerations. This subprogram also computes and displays similarity scores indicating degree of match, in a manner similar to that employed by the BROWSE command, eg:-

QSEARCH string	String in Results Set	Similarity Score %
692	34:692	100
692	3 21.692	0
692	692.416	100
692	693	66
692	6 21.686 (42)	33

The use of the QSEARCH command is expanded in the user manual (Appendix 1). QSEARCHing can be used indefinitely to produce indefinite refinement of a results set, eg:-



The user can set a threshold applying to the above scores of similarity, so that only results above a certain value are included in the results set coming from a QSEARCH. Like the normal SEARCH, details of QSEARCHES are retained for the user's later reference on computer-stored "notepads". Perusal of these details is a simple process, as can be seen from the examples contained in the User Guide - Appendix 1.

QSEARCHing provides a means of UDC string interrogation that the librarians had only previously experienced by way of lengthy manual browsing, and so it was viewed as a powerful aid in the reduction of recall and refinement of relevance. In fact, QSEARCHing for separate facets was deemed so important that, as a result of early trials, provision was made for this process to be performed with much greater efficiency than string by string comparison allowed, This was done by

breaking up all the original UDC keys into their separate facets and storing them on a new B*-tree, thus enabling the user to search for the occurrence of an individual facet anywhere within any string with the fast access speed associated with the tree, the relevant command being the:-

FSEARCH command:- (i) user supplies a UDC code that should only contain a single facet, eg. 34,(42), "1963" etc.

(ii) computer searches the "Facet Btree" for the supplied UDC code.

(iii) search details are reported to the user and stored for later reference.

Following this, the user can BROWSE through the Facet Btree, starting in the results set provided by the FSEARCH. A QSEARCH can follow on FSEARCH just as it can follow an ordinary SEARCH.

A comprehensive set of manuals - contained herein as Appendices 1 & 2 - were made available to the librarians in order to introduce them to computers in general, and the Film Library system in particular. The complete retrieval system (program "RETRIEVE") is described in APPENDIX 1.

So far I've made little mention of the system for updating the files (program "UPDATE"). This separate facility contains the full range of B*tree software for insertion and deletion, but the user interface is extremely simple, and other than the program description which follows, the reader is simply referred to the relevant user manual which is included here as APPENDIX 2.

The complete system consists of two separate programs, RETRIEVE (the retrieval program) and UPDATE (the file maintenance program).

(1) RETRIEVE consists of 12 routines (10 COBOL and 2 UNIVAC Assembler) but, thanks to the use of program segmentation, runs in only 18,000 words (36 bits) for both data (4k) and instructions (14k).

The COBOL data area is described in 365 lines of code, and the total number of COBOL sentences in all 10 routines (Procedure Division) amounts to 2,341 lines of code (including comments). The two Assembler routines only occupy 65 lines of code (data and instructions).

(2) UPDATE consists of 7 routines (5 COBOL and 2 Assembler) and, again by segmentation, program size is limited to just over 20,000 words in total (16k instructions and 4k data). The COBOL data area is described in 245 lines of code, and COBOL instructions occupy 2,356 lines. The two Assembler routines take up 52 lines.

Segmentation causes no significant hindrance to either program (in terms of core-to-disc swapping). A new COBOL compiler introduced since I finished work on these programs, would reduce their size yet further (the code it produces is more re-entrant). The Assembler routines are not complicated, and do little to reduce the potential for machine transportability that the use of COBOL was intended to provide. I am forced to the observation that much of the self documentation attributed to COBOL is missing in programs where the algorithmic detail is rather more complex than the:

MOVE TAX TO SALARY DEDUCTIONS

type of statement with which this language is commonly concerned, and so it was necessary to add a certain amount of explicit additional comment documentation, especially in the B-tree manipulation programs. In fact, the use of COBOL as a language for tree operations was not ideal, and something like ALGOL 68 would almost undoubtedly have been a better choice if structured programs were to have been produced. But the file handling ability of COBOL made it a reasonable choice for this property alone, and few would complain at a program size of around 20K, even in an interactive program.

The B* tree routines made possible all of the features outlined in the description of these structures that I gave earlier. A 30 key page size was used (up to 60 character per key) which caused the UDC file (5½K keys) to be held in a tree of height =3.

At a conservative estimate, I would say that if all the routines had been written in an Assembler language, program sizes of nearer 10K for RETRIEVE and 12K for UPDATE would have been possible. What is more, the great amount of page handling in UPDATE was a cause of inefficiency in respect of execution speed, since the pages (COBOL direct access records) were treated as individual arrays of keys (or "tables" as they are known in COBOL), and the handling of indexed variables in this language is certainly less than one might hope.

Before remarking on the reactions of Film Library personnel to the computerized system, a few observations can be made regarding the operational characteristics of the programs:-

(1) Retrieval Speed was very fast. A SEARCH operation in the UDC file (5,564 keys) involved nine page accesses to completely define the limits of the results set (three B* tree searches with height=3), followed by a sequential brother-to-brother

leaf scan within the limits of the results set to glean the necessary details. The leaf scan was the slowest element in this process when a large results set was being dealt with, but even a results set of several thousand keys could be completely analysed within a couple of seconds of "sit and wait" time. This sort of retrieval speed could be maintained for the complete UDC file of 400K items by using a larger page size, or subjected to only slight detriment by keeping the same page size (to reduce storage overheads) but increasing the tree height to 5 - resulting in a 40% reduction of retrieval speed. The predictability of a balanced tree is part of its beauty.

(2) Updating the files (on-line) was also quick and efficient, and again unlikely to suffer from scaling up to any great extent.

Initially the programs were explained to two senior librarians, and it was via one of these worthy individuals (the Head of Cataloguing and Classification) that the training of some 20 other librarians was undertaken. It was considered beneficial that the staff should be introduced to the computer by a librarian, whose mode of expression would be readily understood. The training librarian also produced a brief set of notes to augment the user guides. Librarians were trained in pairs for about two hours at a computer terminal, and they were left to do all the typing themseleves. Each pair was then issued with a question sheet which they were encouraged to work through in allocated time slots on the computer. Meetings were then held to gauge the general reaction of the librarians to the computer system.

The implementation of the retrieval system was on the UNIVAC 1121 computer at the Open University in Milton Keynes. Access to the machine by the librarians at Brentford was via the GPO telephone network working through MODEMs, which led to a certain amount of line interference. This was particularly irksome, since this group of first-time computer uses seldom knew whether it was their fault or that of the telephone system when their sign-on password failed to be accepted by the computer. Once connected, noise on the telephone line often corrupted retrieval system commands, and although some software was developed to filter out the worst of this interference, there was still enough noise on the line to produce errors in correctly typed commands, thus undermining confidence. In a production system, a hard-wired line would remove this problem completely.

However, in the short amount of time each librarian used the system (under 4 hours in total), the degree of progress was, on the whole, phenomenal. Although a residue of uncertainty remained in most of their minds over one or two aspects of the system (such as the more complex QSEARCH operations) the general level of understanding was high, and the more simple procedures were completely grasped almost at once. The speed with which this understanding was reached, was primarily due to the parallelism between computer and manual systems, and it was only where the operation of the two systems diverged - QSEARCH, FSEARCH etc. that difficulties were encountered. (The librarians' familiarity with typewriter keyboards was also of assistance).

Due to Film Library workload, the trial period was much shorter than one would have hoped, but in the debriefing meetings that followed the trails, the following changes to the computer system were frequently suggested:-

(i) Certain elements of the RETRIEVE program should be re-constructed into a simple stepwise retrieval process of great specificity, that could be used to answer run of the mill enquiries without recourse to any of the more subtle commands.

(ii) Rather than having to type commands, keys should be dedicated to specific commands. For example, there could be a TERM key and a SEARCH key.

(iii) It appeared that a certain amount of typing was unnecessary, and that this should be eradicated.

Although reaction varied greatly, the underlying current of interest in computer methods was encouraging. There were exceptions to this of course, some purely personal worries were expressed, but others were bothered by more solid objections to the sociological effects of computer takeovers, and the inevitable repercussions on job security.

Summary

Operating the programs at the Film Library served as a useful introduction of the librarians to the capabilities of the computer. The ease of file maintenance and freedom from filing errors was considered a major advantage, since this at present constitutes a most tedious manual chore. Although the full range of RETRIEVE and UPDATE software was used at one time or another - and would of course be needed (with additions) in any operational system based on the UDC - it was felt that individual sections at the Film Library should have specific subsets of the programs in order to fulfil the majority of their requirements. The possibility of interacting with the computer and the user (over the telephone) at the same time was found particularly exciting.

A UDC based information storage and retrieval system was thus successfully implemented, and the expected advantages of computerization realized. Reliance upon the UDC led to predictability of retrieval performance (from the library point of view) and a degree of rapid assimilation of technique by the librarians that could not otherwise have been expected. Such a system could be regarded as a mainstay of a more complex retrieval process making use of less conventional techniques, such as those described in the following chapters.

CHAPTER 6

A Learning System

This chapter describes an Information Retrieval System - initially UDC based - capable of assimilating indexing information from the user, and thereby rendering itself more hospitable to the user's mode of expression.

The System Vocabulary (Subject Index, Keyword list or whatever), the common interface which must exist between user and retrieval system, is that at which the language comprising the query is made amenable to, or is "understood" by, the system. Ideally, it will be the system that is made to understand, rather than the user being compelled to couch her query in certain terms. The obvious solution is to have the biggest possible vocabulary made available in anticipation of the retrieval system, but this has its drawbacks in respect of storage and coverage, as even the largest of vocabularies will doubtless have omissions. A better solution would be to have a system that can dynamically expand its system vocabulary, from which it follows that an improvement in indexing becomes possible. One would like to see such improvements applied dynamically throughout the data collection, rather than to just those items in respect of which the original vocabulary expansion occurred, thus constituting a genuine re-indexing step that would preferably be self-implementing.

The specification that emerges, is for a system in which deficiencies in the system vocabulary can be:-

(a) registered,

(b) implemented, and

(c) extended to cover other items to which they might apply.

The first obvious approach is for a human supervisor to make the required changes, by periodically updating files according to the most recent discovered deficiencies. This is the most commonly used solution, and an existing clerical procedure at the Film Library, whereby "query forms" are completed and returned to the Classification Section when errors are spotted, is one example of it. However, the more steps that are involved in a correction procedure, the less likely it is that the procedure will be followed (or even initiated), so one immediately appreciates the desirability of a system in which deficiencies are registered, and even corrected, at the original user/system interface where they are first encountered.

Before coming to a retrieval system, the user has most probably framed a query in her mind. This "raw query" is unconstrained, in that it has yet to be hammered into a shape that the system can understand (i.e. put in terms of its own vocabulary). It is therefore in the inability of the system to grasp the raw query that the first, and possibly the most dangerous, defect exists, since it usually provokes a compromise of the user (query reformulation) rather than an improvement of the system (vocabulary expansion). In my opinion however, this should be, as near as possible, completely reversed, provided that acceptable retrieval efficiency can be maintained, since only then can useful expansion be

provided, long term efficiency maintained, and user hospitality -- vastly and continuously improved. One has in the "raw query" an extremely valuable expression of the required material, and although it will often be inadequate, misinformed or even absolutely useless, it still has the potential to uprate the system according to the best possible motive, namely user needs.

Generally speaking, because of their inherent characteristics, natural language systems offer more obvious possibilities by way of self-implementing improvement (see Chapter 7), but as I go on to show below, computerization makes such dreams possible for more conventional schemes, notably the UDC.
It is worth observing that the more naive the user, the more "raw" is the query and, therefore, potentially at least, the more valuable. In other words, users will gradually become subconsciously constrained by even the most benign system, according to their previous experience.

It is in analysing the raw query to which I alluded above, that a system is effectively able to learn from the user.
This analysis has two components:-

(1) Extract those concepts from the raw query with which the system is already familiar - these will be used in the retrieval search.

(2) Isolate the remaining concepts in the raw query - these might be used to expand the system vocabulary, in which case they are known as "auxiliary terms" (i.e. terms that grow in addition to the original vocabulary).

The concepts in group (2) can fall into one of three classes, they can be:-

(i) synonymous with existing components, or

(ii) concepts omitted from the system vocabulary, but present in the collection (film items), or

(iii) concepts omitted from the system vocabulary not present in the collection.

Concepts of types (i) and (ii) constitute inexcusable defficiencies, and should therefore be assimilated with top priority (thus becoming "auxiliary terms"). Intuition would indicate that concepts of type (iii) should not be considered further, but there is some justification for registering all the new concepts that a user raises - an expanded vocabulary doubtless leads to greater confidence in a system, since one can then be assured that the system knows of the concept, even though it contains no material concerning the concept. However, assimilation of type (iii) concepts is potentially dangerous, since a retrieval system must, when all's said and done, reflect material content rather than omission. For this reason, the practical system described below effectively filters out and rejects type (iii) concepts, thus encouraging the user to home-in on available material, rather than be led up the garden path.

At this juncture, I must re-state the points made above, but now distinctly interms of a UDC based system:

As our language changes, and as subjects vary their relationships one to the other, anything less than the most responsive of systems cannot keep up with the terminology of the user, and the latter is forced to restrain his expectations of a system that can only slowly adapt. This sluggishness on the part of the system can

make itself felt in two ways:-

(i) The Subject Index fails to contain (ie. understand) new words. When the words are well-known synonyms this is unlikely to pose an intractable problem, but when shades of meaning differ, the user could remain unaware of those synonymous linkages which constitute the subtle variations of man's descriptive capacity.

(ii) Subject access to individual documents doesn't change from the time they are initially entered in the system, although their subject matter might passively vary, due to outside circumstances, as time goes by. In other words, the changing patterns of knowledge and the ebb and flow of relationships between subjects, demands that a measure of re-indexing be used to trace the time dependent juxtapositioning of concepts.

The Subject Index clearly emerges as the villain of the piece, since it is here (otherwise known as the "term vocabulary") that all the variable parameters by which the total information collection is defined, are registered. So my aim was to produce a form of computerized UDC system that could, at the same time as performing efficient retrieval, mould and reshape the Subject Index into a more flexible "term" or "system vocabulary", according to what could be learnt from the user.

The approach that I adopted in a program called "LEARNER" is summarized in figure 6.1, and is now described in some detail:-

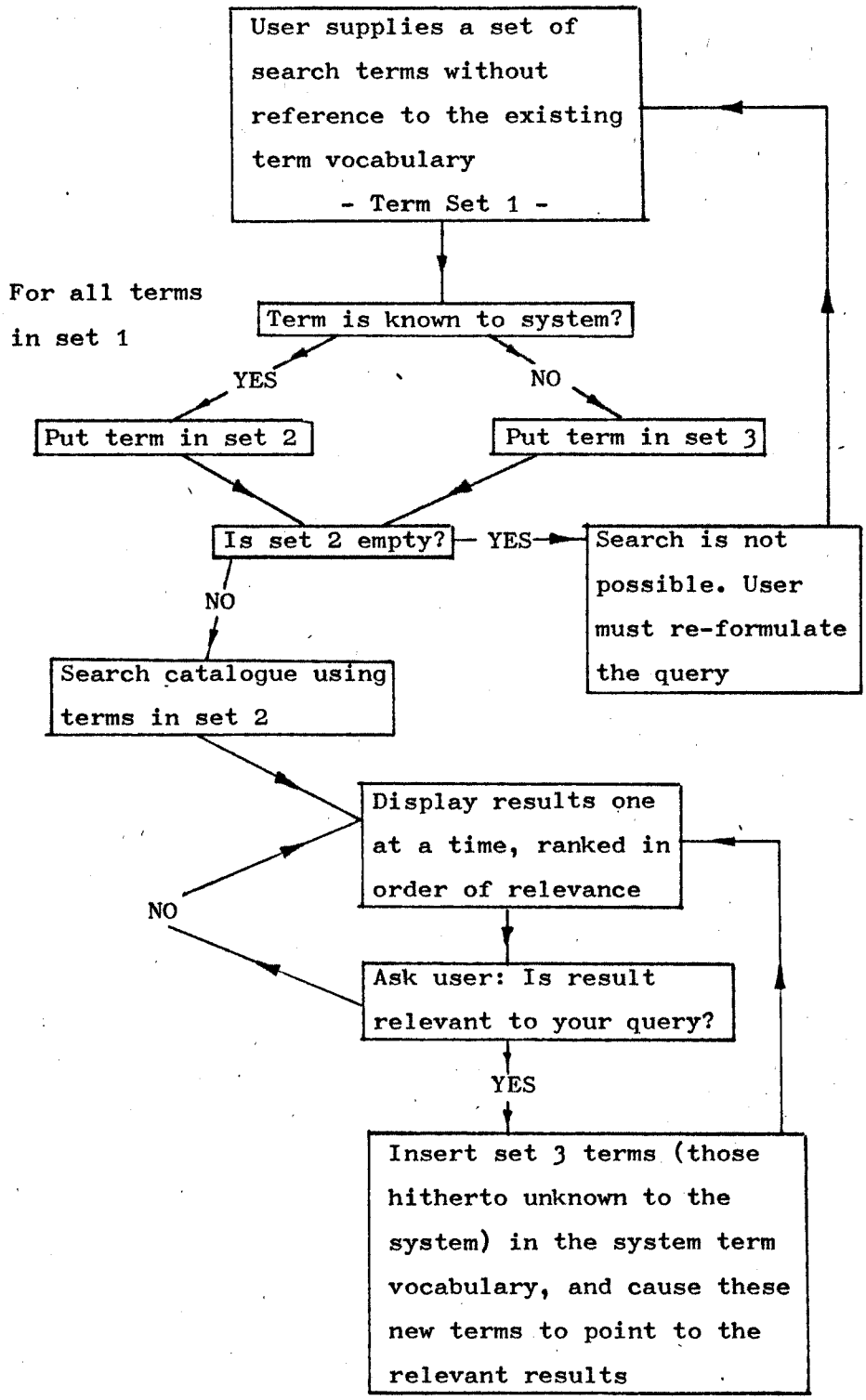


figure 6.1 - The LEARNER System

(1) The user types in the basic query, by way of the concepts with which the retrieved item must be concerned. This constitutes the "raw query". No reference is made to the Subject Index until this basic set of requirements has been typed, thus ensuring that the user is not influenced by the limitations of the system.

(2) Using the raw query, the system searches the Subject Index and finds the closest alphabetical match to each of the terms that the user typed in. The user can browse around the Subject Index, but eventually she must respond for each term with one of the following statements:

YES - this subject is one I wish to use in the search.
NO - there is no entry in the Subject Index that satisfies this element in the raw query.
- scratch this element from the raw query.

In making these simple responses, the user does in fact comment on the sufficiency of the Subject Index. A "YES" means that the Subject Index has understood this part of the raw query, whilst a "NO" might indicate a conflict of understanding (it might also indicate the purely physical absence of film on a given subject, but no retrieval system can compensate for missing items). Unfortunately, the "NO" response - which is the essence of the learning system, because it is here that defficiencies are registered - is also the potential source of system abuse, since it should only really be issued when the user is positive about the inadequacy of the Subject Index in respect of the current term - more of this later.

(3) A search of the main UDC file is instigated, based on the search parameters already selected and confirmed ("YES" responses).

(4) The results of the search are displayed, ranked in the order with which they satisfy the search parameters. If a document is chosen (i.e. the user deems the search successful in respect of a given document) then then the following action is taken by the system:

(i) The final search parameters are compared with the raw query. If they match (i.e. there weren't any "NO" responses) no further action is taken.

(ii) If there were some "NO" responses, then the elements in the raw query that could not be satisfactorily matched in the Subject Index are now themselves entered in the Subject Index, and made to point to the selected documents. In other words, it is assumed that a connection exists between the result and the raw query, even though a particular term in the raw query was not in any way involved in the search process.

Subsequent queries will now draw on an expanded Subject Index, containing both UDC entries and those "auxilliary terms" that have resulted form failure to match raw queries. These auxilliary terms point directly to the items with which connections have been established, and whenever they are used in a successful search, a relationship between term and chosen document is consolidated, even though the term might not have been responsible for retrieving the chosen document (again a connection between raw query and acceptable results is assumed).

The system of auxiliary terms relies on a weighting scheme for both terms and the documents to which they point, with the result that the usefulness of a given auxiliary term, and the validity of a connection between a term and a given document, is only slowly inferred.

Before going on to give more detail on LEARNER (that is, the inferential steps by which user-implied connections are described) I shall give a practical example in which the printout is followed by descriptive notes. First, I must stress however, that in a production system, search results would consist of full card prose (and not simply the UDC descriptors) since only by reference to this prose would useful item selection be possible.

Example:-

QUERY : NATO + Cod War + Trawlers

Note! Lines involving user input are denoted by the > symbol in the first character position.

```
>@XQT FILMX.LEARNER

ENTER COMMAND
>INPUT
ENTER TERM (# TERMINATOR)
>COD WAR
>NATO                                    (1)
>TRAWLERS
>#
        COD WAR WITH ICELAND
        341.225.1 (491.1)
>YES                                     (2)
        N.A.T.O.
        341.232.1 N.A.T.O.
>YES                                     (3)
        TREASON
        341.356
>N
        TREATIES
        341.24                                   (4)
>J-2
        TRANSKEI INDEPENDENCE
        341.231 (685.4 TRANSKEI)
>NO
```

```
ESTIMATED SEARCH TIME = 0013 SECONDS FOR SCORED SEARCH
ESTIMATED SEARCH TIME = 0013 SECONDS FOR EXACT SEARCH
000    1=0000 2=0204 3=0272

ENTER COMMAND
>SEARCH
RESULTS FILE FILLED BY TERM:- N.A.T.O.

ENTER COMMAND
>DISPLAY
  SCORE=020  UDC= 341.225.1(491.1)341.652:341.232.1 NATO(491.1)
> .....carriage return
  SCORE=020  UDC= 341.225.1(491.1)341.652:341.232.1 NATO(491.1)
>YES
SELECTED ITEM NO:- 1176
  SCORE=020  UDC= 341.225.1(491.1)341.652:341.232.1 NATO(491.1)
>  c/r
  SCORE=020  UDC= 341.225.1(491.1)341.652:341.232.1 NATO(491.1)
>YES
SELECTED ITEM NO:- 1174
  SCORE=010  UDC= (491.1 KEFLAVIK) 341.232.1 NATO
>  c/r
  SCORE=010  UDC= (491.1 GRINDAVIK) 341.232.1 NATO
>@
END OF EXECUTION
```

```
>@XQT FILMX.LEARNER

ENTER COMMAND
>INPUT
ENTER TERM (# TERMINATOR)
>COD WAR
>TRAWLERS
>#
      COD WAR WITH ICELAND
      341.225.1 (491.1)
>YES
      TRAWLERS
>YES
ESTIMATED SEARCH TIME = 0000 SECONDS FOR SCORED SEARCH
ESTIMATED SEARCH TIME = 0000 SECONDS FOR EXACT SEARCH
000    1=0002 2=0272

ENTER COMMAND
>SEARCH
RESULTS FILE FILLED BY TERM:- COD WAR WITH ICELAND

ENTER COMMAND
>DISPLAY
  SCORE=012  UDC= 341.225.1(491.1)341.652:341.232.1 NATO(491.1)
>YES
SELECTED ITEM NO:- 1176
  SCORE=012  UDC= 341.225.1(491.1)341.652:341.232.1 NATO(491.1)
>  c/r
  SCORE=010  UDC= 341.225.1(491.1)
```

Notes - see circled numbers

(1) User inputs query without reference to the system vocabulary - Set 1 (see figure 6.1)

(2) LEARNER knows about "COD WAR". User affirms understanding by answering "YES", and this term is put in Set 2.

(3) Ditto (2) for "NATO" - Set 2.

(4) LEARNER is unaware of term "TRAWLERS" (user browses around to make sure that it is not mis-spelt) so this term is put in Set 3 and retained when the user answers "NO".

(5) LEARNER estimates search times (the details of this are not important). User decides to go ahead with SEARCH. LEARNER uses terms in Set 2 (i.e. those it already knows about) for searching.

(6) User is shown search output, ranked according to the degree with which the original query has been satisfied. Carriage return causes no action to be taken, but a "YES" response causes the current item to be selected. Also, if a "YES" response is made and Set 3 is not empty, the "learning" process is invoked, causing:-

(i) the term(s) in Set 3 (i.e. TRAWLERS) to be inserted in the system vocabulary, and

(ii) the new system vocabulary term(s) to be made to point to the selected item(s).

(7) Another user comes along.

(8) Now the term "TRAWLERS" is understood (it doesn't have an associated UDC number, thus indicating that it is an auxilliary term - i.e. one supplied by a previous user). Both terms are put in Set 2, and the SEARCH process consists of ANDing the UDC items for "COD WAR" with the previously user-selected "TRAWLERS" items.

(9) The user is shown the search output. If a "YES" response is made to an item retrieved with the aid of the auxilliary (i.e. non-UDC) term "TRAWLERS", then the usefulness rating (i.e. SCORE) of this term is uprated. Also, the SCORE of the particular item is uprated no. 1176 in this case).

LEARNER was written in COBOL. It makes extensive use of B-trees for internal data structures, such as in the System Vocabulary, which must be amenable to rapid updating. The search method is essentially that of ANDing results from the separate components of the original query, and ranking the output in the order in which the totality of the query is satisfied, eg:-

QUERY concerns subjects : A, B and C
LEARNER assumes : A + B + C is required
Ranking - 1st : A + B + C
2nd : A + B OR B + C OR A + C
3rd : A OR B OR C

In this way, all items concerning any of the search components are retrieved, but highest display priority is given to those items that satisfy more than one query component.

Intercession between the user and the system is made by the system vocabulary, which consists of two parts:

(i) the UDC Subject Index, and

(ii) the auxilliary terms gleaned from previous users

Query terms to be found in the Subject Index pose no problem. Terms of the 'auxilliary term' variety need special attention, as do new terms (those hitherto unknown to the system in any guise) which will be entered in the system vocabulary if they are used in a successful search. It is important to note that the user needs make no distinction between the two different vocabulary components - LEARNER blends them into one entity.

All UDC Subject Index entries are given a fixed "value rating" (10 to be precise). The value rating of an auxilliary term is not fixed, but is increased (to a maximum of 10) everytime it is used to produce successful results. Also, the individual items referred to by an auxilliary term are themselves given a value rating, and this is increased every time the item is selected as a successful search result. This latter rating is of particular value, since it reflects a critical judgement under a high level of individual scrutiny. It is the combination of these value ratings that is used to provide the "SCORE" in the above example, and it is this SCORE upon which output ranking is based.

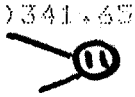
To illustrate SCORE adjustment, the following example charts the repeat of an earlier search:-

```
>@XQT FILMX.LEARNER

ENTER COMMAND
>INPUT
ENTER TERM (# TERMINATOR)
>COD  WAR
>TRAWLERS
>#
      COD WAR WITH ICELAND
      341.225.1 (491.1)
>YES
      TRAWLERS
>YES
ESTIMATED SEARCH TIME = 0000 SECONDS FOR SCORED SEARCH
ESTIMATED SEARCH TIME = 0000 SECONDS FOR EXACT SEARCH
000   1=0002 2=0272

ENTER COMMAND
>SEARCH
RESULTS FILE FILLED BY TERM:- COD WAR WITH ICELAND


ENTER COMMAND
>DISPLAY
  SCORE=014  UDC= 341.225.1(491.1)341.652:341.232.1 NATO(491.1)
>YES
SELECTED ITEM NO:- 1176
  SCORE=013  UDC= 341.225.1(491.1)341.652:341.232.1 NATO(491.1)
>YES
SELECTED ITEM NO:- 1174
  SCORE=010  UDC= 341.225.1(491.1)
>YES
SELECTED ITEM NO:- 1137
  SCORE=010  UDC= 341.225.1(491.1)
>
  SCORE=010  UDC= 341.225.1(491.1)
>
  SCORE=010  UDC= 341.225.1(491.1)
>YES
SELECTED ITEM NO:- 1134
  SCORE=010  UDC= 341.225.1(491.1)
>@
END OF EXECUTION
```



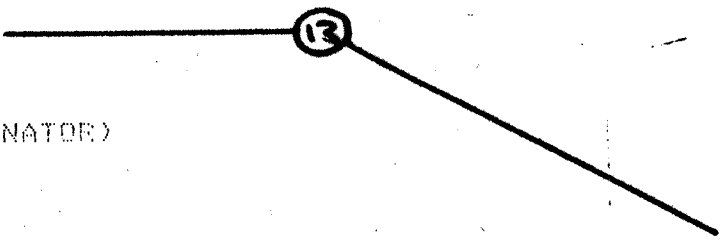
```
>@XQT FILMX.LEARNER

ENTER COMMAND
>INPUT
ENTER TERM (# TERMINATOR)
>TRAWLERS
>#
      TRAWLERS
>YES
ESTIMATED SEARCH TIME = 0000 SECONDS FOR SCOPED SEARCH
ESTIMATED SEARCH TIME = 0000 SECONDS FOR EXACT SEARCH
000    1=0004

ENTER COMMAND
>SEARCH

ENTER COMMAND
>DISPLAY
  SCORE=007  UDC= 341.225.1(491.1)341.652:341.232.1 NATO(491.1)
>
  SCORE=006  UDC= 341.225.1(491.1)341.652:341.232.1 NATO(491.1)
>
  SCORE=003  UDC= 341.225.1(491.1)
>
  SCORE=005  UDC= 341.225.1(491.1)
>

ENTER COMMAND
>@
END OF EXECUTION
```



Notes - see circled numbers

(10) Another user is interested in "COD WAR TRAWLERS". The following search combines "COD WAR" entries retrieved via the UDC Subject Index, with "TRAWLERS" entries retrieved via the auxilliary term component of the system vocabulary.

(11) The first items displayed are those satisfying both search components, and the SCORE is made up of:

10 (COD WAR retrieved via UDC)

+ weighting given to term "TRAWLERS"

+ weighting given to individual item

When the two items satisfying both components are selected ("YES" responses), both the "TRAWLERS" weighting and the weightings of the individual item (1176 and 1174) are increased. These increased weightings will show up as higher SCORES in subsequent retrievals.

(12) Items 1137 and 1134 were not hitherto connected to "TRAWLERS" (they were retrieved via the UDC for "COD WAR" alone), but LEARNER, in assuming a connection to exist between query and successful results, now establishes such a connection. This infact constitutes a re-indexing step, and its application becomes apparent in (13) below.

(13) When retrieval via the auxilliary term "TRAWLERS" alone is attempted, the items connected to this term by previous searches (solely on LEARNERS initiative) are retrieved. The first two connections were established at stage (6), the second two at stage (12).

To summarize, LEARNER has the ability:-

(i) to provide a dynamic system vocabulary (by adding auxilliary terms),

(ii) to automatically undertake dynamic re-indexing (by linking auxilliary terms to items hitherto indexed solely by UDC), and

(iii) to act as a potential component of a natural language retrieval system (see next chapter)

Further work remains to be done in tackling the following points:-

(1) There is no check on mis-spellings. A user could (and undoubtedly would) provide LEARNER with a host of "noisy" auxilliary terms due to mis-spelt query components that were not refined by browsing in the system vocabulary. The weighting system exists to combat this, but common mistakes could conspire to produce a disruptive dichotomy in the system.

(2) The same thing applies to synonyms. If a query term is not matched at first by the existing system vocabulary, the user should look at the more obvious synonyms (plurals etc.) before answering "NO" at the term definition stage. In the previous example for instance, one needed to be sure that neither TRAWLER nor TRAWLERS (nor perhaps even TRAWLING) existed, before faulting the vocabulary.

Again, the weighting system helps here but, what is more important, part of the purpose of LEARNER is to provide synonyms that were previously unanticipated, so the user should not be expected (or indeed encouraged) to go too far in eliminating synonyms, particularly those of the more complex variety (eg. TRAWLERS = FISHING BOATS etc). However, this laissez faire attitude does have its dangers, such as in the following example. Here the user selects an unknown synonym - "SPACE VEHICLE" - whilst LEARNER only knows about such terms as "CAPSULE", "ROCKET", "SATURN" etc. The result is that only the two items selected by the term originator - 4798 and 4795 - are connected to this term.

```
ENTER COMMAND
>INPUT
ENTER TERM (# TERMINATOR)
>SPACE VEHICLE
>BLAST OFF
>#
      SPANISH
      (=51)
>N-
      SPACE TREATY, EAST WEST
      341.24 : 629.19
>N-
      SOUTH EAST ASIA TREATY ORGANIZATION
      341.232.1 S.E.A.T.O.
>NO
      BLAST OFF, ROCKETS (*)
      M74 : 629.192
>YES
ESTIMATED SEARCH TIME = 0000 SECONDS FOR SCORED SEARCH
ESTIMATED SEARCH TIME = 0000 SECONDS FOR EXACT SEARCH
000   1=0000 2=0155

ENTER COMMAND
>SEARCH

 ENTER COMMAND
 >DISPLAY
   SCORE=010  UDC= M74 : 629.192
 >
   SCORE=010  UDC= M74 : 629.192
 >
   SCORE=010  UDC= M74 : 629.192
 >
   SCORE=010  UDC= M74 : 629.192
 >
   SCORE=010  UDC= M74 : 629.192
 >YES
 SELECTED ITEM NO:- 4790
   SCORE=010  UDC= M74 : 629.192
 >
   SCORE=010  UDC= M74 : 629.192
 >
   SCORE=010  UDC= M74 : 629.192
 >YES
 SELECTED ITEM NO:- 4795
   SCORE=010  UDC= M74 : 629.192
 >
   SCORE=010  UDC= M74 : 629.192
 >
   SCORE=010  UDC= M74 : 629.192
 >
   SCORE=010  UDC= M74 : 629.192
 >@
 END OF EXECUTION
```

A subsequent user then comes along, and simply asks to see items concerning "SPACE VEHICLES". This trusting enquirer is then only shown two items, whereas she should really see all of the items on "CAPSULES", "ROCKETS" etc. as well.

```
ENTER COMMAND
>INPUT
ENTER TERM (# TERMINATOR)
>SPACE VEHICLE
>#
      SPACE VEHICLE
>YES
ESTIMATED SEARCH TIME = 0000 SECONDS FOR SCORED SEARCH
ESTIMATED SEARCH TIME = 0000 SECONDS FOR EXACT SEARCH
000   1=0002

ENTER COMMAND
>SEARCH

ENTER COMMAND
>DISPLAY
  SCORE=002  UDC= M74 ; 629.1^2
>
  SCORE=002  UDC= M74 ; 629.192
>

ENTER COMMAND
```

Of course in reality, all but the most naive of users would themself think of synonyms if their first search proved inadequate , and, in time at least, the new term "SPACE VEHICLES" would be judged by its performances, and either consolidated into, or dropped from, the system vocabulary. What is needed therefore, is some simple warning message that would guard against over-confidence in new terms, but at the same time (and this is important to the systems continued performance improvement) allow LEARNER to absorb all the novelty of expression that is on offer.

Any unease that one might have felt in respect of the above disadvantages has a simple remedy, though it is most certainly a case of "treating the symptoms", since it involves using LEARNER as a system of suggestion rather than as one of direct implementation.

If LEARNER is allowed to make no system vocabulary updates of its own accord, but is instead used to alert a supervising librarian of necessary alternations, then one's worries disappear, but only together with a measure of automation. In this manner, LEARNER could be used to act upon the Special Schedules of the UDC kept at the Film Library, and not upon the Subject Index itself (which would remain inviolable to direct tampering). The intellectual worth of LEARNER would not be diminished, but its self-supervising ability would be lost. Seen in this guise, LEARNER has achieved enough acceptance in trials at the Film Library to be considered for operational implementation.

However, caution apart, the potential of this type of system is hard to ignore, and I think that relegation of LEARNER to a menial role, comprising little more than the logging of oddities, would constitute a travesty of the heightened automation that a computer is expected to provide. Rather than making LEARNER subservient to a supervising librarian, I would prefer to see it operating as it stands, but incorporating a far greater reticence to introduce new terms to the system vocabulary - a step simply arranged by adjustment of the weighting strategy. Actually, a more critical weighting scheme might be extended to take in UDC classifications, thereby comprising a confidence measure of the original classification. (At present, all UDC retrievals are arbitrarily valued at 10.)

It was not possible to subject LEARNER to the same scrutiny under trial at the Film Library as that extended to the RETRIEVE and UPDATE programs described in the previous chapter. Tests on a smaller scale however, and interactions with a number of librarians, showed that the ability to usefully trap the user's mode of expression ("useful" that is, in relating the unconstrained query to the items finally chosen by the user) was indeed manifest. Although the weighting strategy was not strict enough to restrain unruly growth of the vocabulry, it was clearly seen (by monitoring SCORE variations) that regulation could be meaningfully imposed via weightings, and that by using the "watered down" version of the program (a system of suggestion rather than of direct action) the awareness of user expression could be reflected in the system interface (the modified Subject Index) thus maintaining hospitality.

In fact, even without its learning capability, LEARNER offers a neater (though less comprehensive) method of straight UDC searching than the pilot RETRIEVE program. A simple post-coordinated search facility after the LEARNER fashion (the coordinates being the individual elements of the raw query) could offer a useful addition to any proposed computer armoury in the Enquiries Section, and in closing this chapter, I shall give a couple of examples of LEARNER being used to perform coordinated UDC retrieval searches, via an unadulterated Subject Index:-

```
ENTER TERM (# TERMINATOR)
>COD WAR
>REYKJAVIK
>#
      COD WAR WITH ICELAND
      341.225.1 (491.1)
>YES
      REYKJAVIK
      (491.1 REYKJAVIK)
>YES
ESTIMATED SEARCH TIME = 0010 SECONDS FOR SCORED SEARCH
ESTIMATED SEARCH TIME = 0010 SECONDS FOR EXACT SEARCH
000   1=0038 2=0272

ENTER COMMAND
>SEARCH
RESULTS FILE FILLED BY TERM:- COD WAR WITH ICELAND

ENTER COMMAND
>DISPLAY
  SCORE=020  UDC= 341.225.1(491.1)322.233(491.1 REYKJAVIK)

>
  SCORE=020  UDC= 341.225.1(491.1)07.06(491.1 REYKJAVIK)
>
  SCORE=020  UDC= 341.225.1(491.1)061.3(491.1 REYKJAVIK)
>
  SCORE=020  UDC= 341.225.1(491.1)061.3(491.1 REYKJAVIK)
>
  SCORE=020  UDC=.341.225.1(491.1)061.3(491.1 RE/KJAVIK)
>
  SCORE=020  UDC= 341.225.1(491.1)061.3(491.1 REYKJAVIK)
>
  SCORE=020  UDC= 341.225.1(491.1)061.3(491.1 REYKJAVIK)
>SCRATCH

ENTER COMMAND
>INPUT
ENTER TERM (# TERMINATOR)
>JEWS
>ARABS
>#
      JEWS
      (=924)
>YES
      ARABS
      (=927)
>YES
ESTIMATED SEARCH TIME = 0012 SECONDS FOR SCORED SEARCH
ESTIMATED SEARCH TIME = 0012 SECONDS FOR EXACT SEARCH
000   1=0101 2=0122

ENTER COMMAND
>SEARCH
RESULTS FILE FILLED BY TERM:- ARABS

ENTER COMMAND
>DISPLAY
  SCORE=020  UDC= (=927)343.435 (=924)(42)
>
  SCORE=010  UDC= (=924)(759 MIAMI) 581.52
>
  SCORE=010  UDC= (=924)94.542.1 (438)
```

Document Clustering and Associated Experiments

Figure 1.5a showsa BBC Film Library catalogue card. In the UDC system, primary retrieval is based solely on the mnemonic code that represents the prose, and the prose itself is only referenced at the last stage of setting film items for despatch. The prose is,however, a rich source of information in its own right, and if the pertinent words could be extracted from it and presented in some suitable form, these words would constitute an immediate characterization of the card, and hence the film it represents. What is more, if two cards are found to have a large number of words in common, one can assume that a good deal of similarity in their filmic counterparts is inevitable, and is therefore provided with the basis of a subject classification founded on document-document similarity that involves the raw material (words) rather than applied descriptors (UDC codes).

This simple idea is the basis of Document Clustering, which has evolved into a major area of information retrieval that despite disappointments (in terms of efficiency) continues to promise high levels of library automation. A good deal of ground work has been done over recent years by SPARCK-JONES, SALTON and many others (see CHAPTER 3).

In this Chapter, some basic experiments in clustering applied to Film Library catalogue cards are described. These experiments are expanded into test sytems for:-

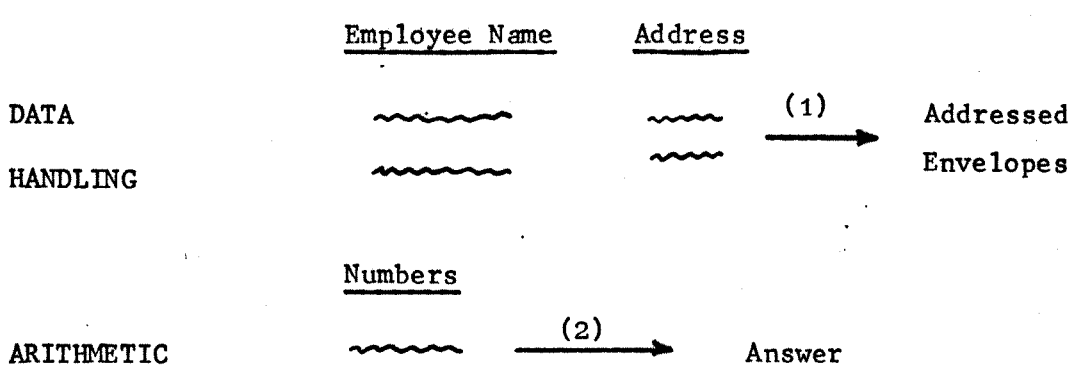
(i) automatic UDC attribution

(ii) natural language retrieval, and

(iii) clustering of specific areas "on demand" as part of a modular system (see CHAPTERS 4 and 8).

The justification for this work, which in terms of clustering technique is not particularly novel, is that it introduces a variety of possibilities which, when conjoined with other, more orthodox, methods, produce yet greater enrichment of the multi-faceted retrieval approach to be discussed in the next Chapter.

Although the original concept of a computer was probably that of a giant calculating machine, the greater part of modern commercial computer usage is concerned with data handling rather than numerical computation. The division between these last two applications need not be straightforward - confusion of disciplines can make comparisons possible that were hitherto neglected - since both can be seen as transformations performed according to stated rules:-



where transform (1) is a computer program, and (2) is known as 'mathematics'.

Prose is the starting point for the librarian. Initial transformations into Keywords, Dewey Classifications or whatever, are methods of circumventing the problems posed when organizing raw language, and, since any such step is subjective, they can be criticized on these grounds alone.

A document (e.g. a catalogue card, abstract etc.) is a set of words. If this set of words is a definitive description of the original (e.g. piece of film, scientific paper etc.) then it can be viewed as being the article itself, implying that the 'set of words' constitutes the absolute information content. If no assumptions are made about words as distinct from numbers or codes, then the following questions must be asked:

(1) Is word order important?

(2) Are words meanings affected by their neighbours?

(3) Are some words more important than others?

(4) Do some words overlap in meaning?

One can go on to ask:

> Is a complete representation of a document possible in a form than can never be faulted? That is, can we define prose mathematically?

The simplest representation of a document in terms of its natural language content alone, is as a point in an N-dimensional space, where N is the number of descriptors whereby the total document collection is defined - the N descriptors comprise the 'system vocabulary'. This point (or N-element vector) thus depicts the relationship between the document and the rest of the collection. Assuming the adequacy of this model, one can say that like vectors will denote like documents, and that's an end to it. But if conditions (1) to (4) above disrupt the ideal, then clearly something more expressive than an N-dimensional space is required.

What is more, given that one would prefer a fully objective system, it must be stipulated that only solid, rule-governed procedures can be used to produce a mathematical definition of prose, because then it is certain that the transforms comprise a genuine 'data arithmetic', and do not disturb its elementary nature (its meaning).

Since questions (1) to (4) above must be answered in the affirmative, one is faced with a formidable number of variables. The once simple vectors become hideously tangled, and in providing a complete data arithmetic for the definition of prose mathematically, an immense task emerges. Yet more important however, is the fact that some of the necessary transformation rules do not seem amenable to objective solutions. How, for example, does the computer infer connections between disparate words? How are variations in meaning registered due to changes in word order, when "meaning" has no mathematical counterpart?

In looking at practical attempts to perform automatic document classification based on free prose, one soon sees that such attempts are forced to circumvent the problems of language rather than confront them fully, and so where theory falters we must enlist empiricism, and its ally, statistics, in order to produce adequate data transformations.

I have identified three reasons why automatic document analysis is not perfectly straightforward, namely synonyms, common words and interrelationships , but if documents are to be organized according to their subject matter, with similar documents being grouped together (i.e. "clustered"), then surely no element of the subject analysis process can be overlooked. Well, this whole area is, to say the least, fluid.

COMMON WORDS can be automatically isolated by considering word (term) frequency distributions, since they show up by appearing with high frequency overall, but with little frequency variation from document to document. If common words are regarded as noise, then obviously a form of noise that remains constant throughout the collection will have no imbalancing effect, and can be subtracted from the

data (filtered out). As to whether all common words can be isolated automatically however, there is some doubt, and one is forced back to ask "what is a common word?". If "the" is rejected as a common word, then a politician called "MR. THE" could constitute a major disruptive element in a hitherto satisfactory retrieval system.

SYNONYMS can be dealt with by means of a thesaurus. Using a machine readable form of a standard manual thesaurus is one approach, but this is a cumbersome solution when one has automation as a watchword. Purely automatically derived thesauri are rare and, to my knowledge, non-existent on a major scale. In the same way that document clustering relies on term correlation, so automatic term clustering (thesaurus construction) relies on document correlation (SALTON[252]).

Suffice it to say that to produce a worthwhile thesaurus automatically, one would need a large stock of data from which to infer connections between terms. In fact, the bulk of data is both the maker and breaker of all clustering techniques, because, on the one hand, a large stock is the best way of overcoming small inconsistencies that could otherwise distort the system, but, on the other hand, bulk is the sure way of reducing system efficiency in terms of speed. Simple synonyms (plurals etc) can be dealt with easily (though with some degree of ambiguity) by relying upon word stems, rather than full words.

WORD RELATIONSHIPS require the attention of linguists rather than statisticians or computers. Consider two phrases:

United man states reasons

and United States man reasons

How is an automatic system to differentiate between them? Just

one word can vary its meaning according to the words which accompany it, e.g.:-

Blow up

Blow football

Blow by blow

Attempts to circumvent this most extraordinary type of confusion are being made, but the less burdensome approach is to ignore it, and hope that things smooth themselves out when averaged over a large number of examples. It is easier (much easier) to isolate simple phrases - e.g.: "blast off", "splash down" - in which word relationships are fixed; in fact the phrases constitute little more than words with spaces in them.

In what follows, the control of synonyms and the removal of common words was purely manual. For the most part, this external control was of a low level, and

apart from one last phase, which relied upon a deal of interpretation to control synonyms, I tried to impose only that degree of human intervention that one could reasonably expect to be programmable; for this reason I took no account of word relationships - not even simple phrases.

An algorithm was written to compare documents (Film Library Catalogue cards) and assess their similarity by way of the number of words they had in common - documents having the most words in common were therefore considered to be the most similar. Comparing each document with every other document in the test collection (which consisted of 39 documents in total) gave rise to a document - document similarity

matrix, from which clusters were derived according to the following strategy:-

If document A is similar to document B
and document B is similar to document C
then A is assumed to be similar to C (ie. 'single link' clustering)

The phrase "is similar" in the above definition, could be read "has a certain number of words in common with", by which I mean that:

Similarity α Number of shared words

If a high similarity threshold was set in the clustering algorithm, then the clusters consisted of very similar documents, and by varying the "threshold similarity" (below which clustering does not occur) I was able to vary the coherence of the clusters. The 39 documents were grouped as follows (according to the UDC) :-

Subject Area	Document Numbers
Car Crashes	1-5
Rocket Launches	6-13
Moroccan Arabs	14,15
Jews, Concentration Camps	16-19, 20-25
Independence (of Africa, Asia & Israel)	26-28, 31-34
Forensic Science	29, 30
Trials	35-37
South American Rivers	38, 39

These documents were clustered under the following succesively higher levels of control:

(1) NO CONTROL. Documents were clustered as given.

(2) COMMON WORDS REMOVED. All occurences of words such as "A", "AN", "AS", "AT" were deleted from the data before clustering. The full list of these words is contained in figure 7.0.

(3) LOW LEVEL SYNONYM CONTROL + COMMON WORDS REMOVED

All conflict between the singular and plural forms of otherwise identical words was removed, such as Accident=Accidents. Other grammatical equivalences were also imposed, such as those linking verb tenses, for example Bomb=Bombing. Also, connections such as Algeria=Algerian & Britain=Briton=British were established.

(4) HIGH LEVEL SYNONYM CONTROL + LOW LEVEL SYNONYM CONTROL + COMMON WORDS REMOVED.

e.g. Jew = Israeli; Bomb = Explode; Soldier = Troops; Death = Funeral = Grave = Coffin.

As I said previously, level (4) does not immediately strike one as being a programmable control.

figure 7.0 - List of Common Words removed when clustering

a	about	above	across	after	against	ago
all	along	also	an	and	are	as
at	away	be	because	been	before	being
between	beyond	both	briefly	by	down	during
each	etc	ever	for	from	has	have
he	her	his	how	in	inside	into
is	it	its	not	now	of	off
on	only	other	others	out	that	the
their	them	then	there	thereby	these	they
this	those	though	thru	through	throughout	
to	up	was	were	what	when	where
which	whilst	who	why	with	within	

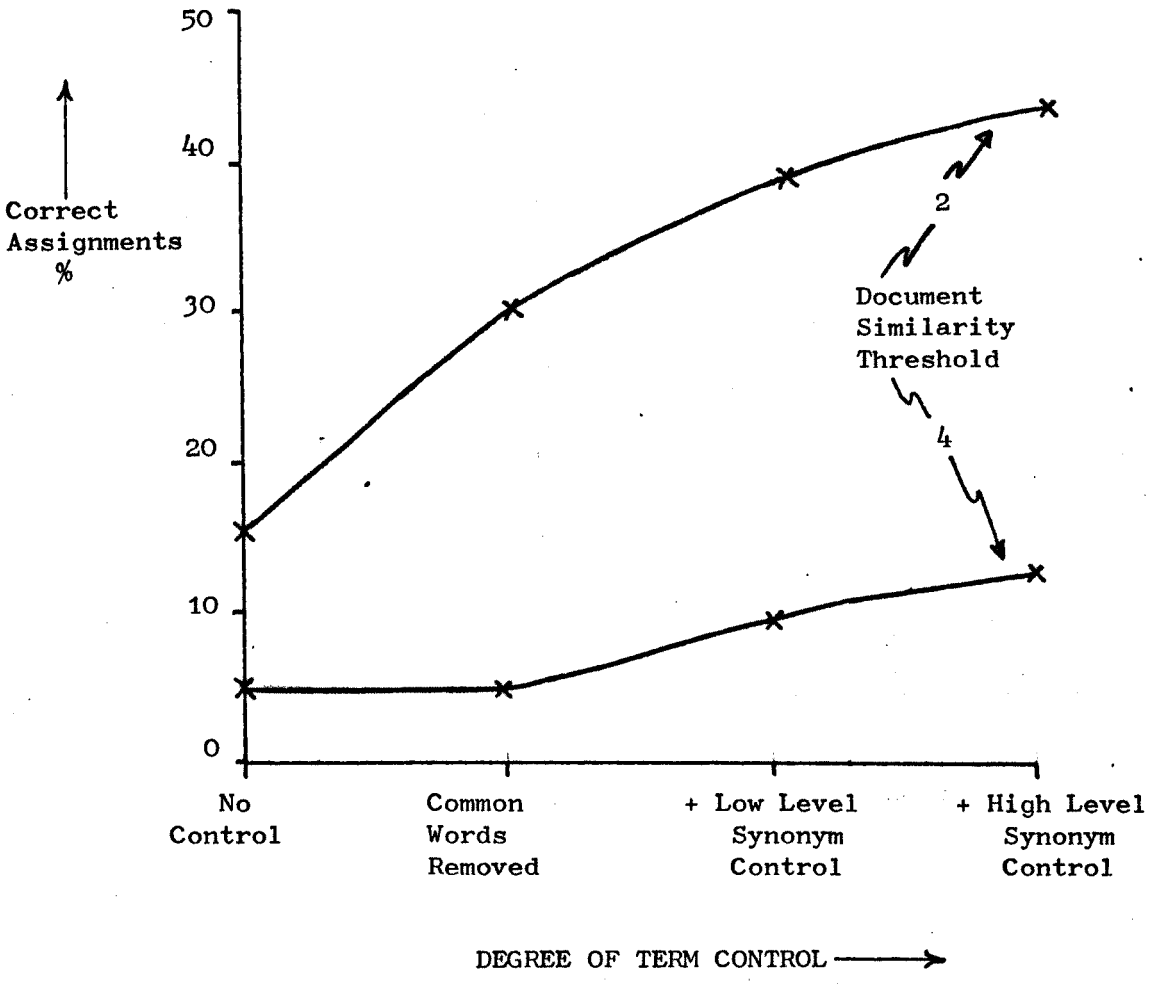
In assessing whether or not a cluster was valid, it was first compared with the UDC classes (see above), but clusters that violated these classes were separately assessed to see if there was any justification for their existence.

Clusters were constructed on the basis of a "threshold similarly" between documents measured on an arbitrary scale (an absolute similarity measure such as "percentage similarity" between documents would have caused document length to become an important factor, for which there is no justification). Clusters containing documents above a similarity threshold of 4 say, are therefore much more coherent than documents in a cluster constructed above a threshold of only 2. The effect of lowering the similarity threshold criterion was thus to encourage the formation of clusters, but also to increase the likelihood of errors.

In fig. 7.1, the percentage of documents correctly clustered (ordinate) is plotted against the degree of control (abscissa) for similarity thresholds of 2 and 4. Attempts to cluster at a threshold lower than 2 led to the production of noisy clusters that could not be justified ("inclusion errors") whilst above this level, documents were either correctly clustered, or not clustered at all ("exclusion errors").

The document collection was then expanded to include 98 documents of which, at a high level of synonym control, 50 were correctly clustered at a threshold of 2.

figure 7.1 - Document Clustering Performance



Rather than simply commenting on these results as they stand, one is able to go below the surface and look at the cause of a particular cluster formation, simply by referring to the word frequency distributions that emerge from automatic document analysis. Consider figure 7.2, which is a plot of (abscissa) the number of different cards on which a given word occurs, against (ordinate) the percentage of different words having a given abscissa value.

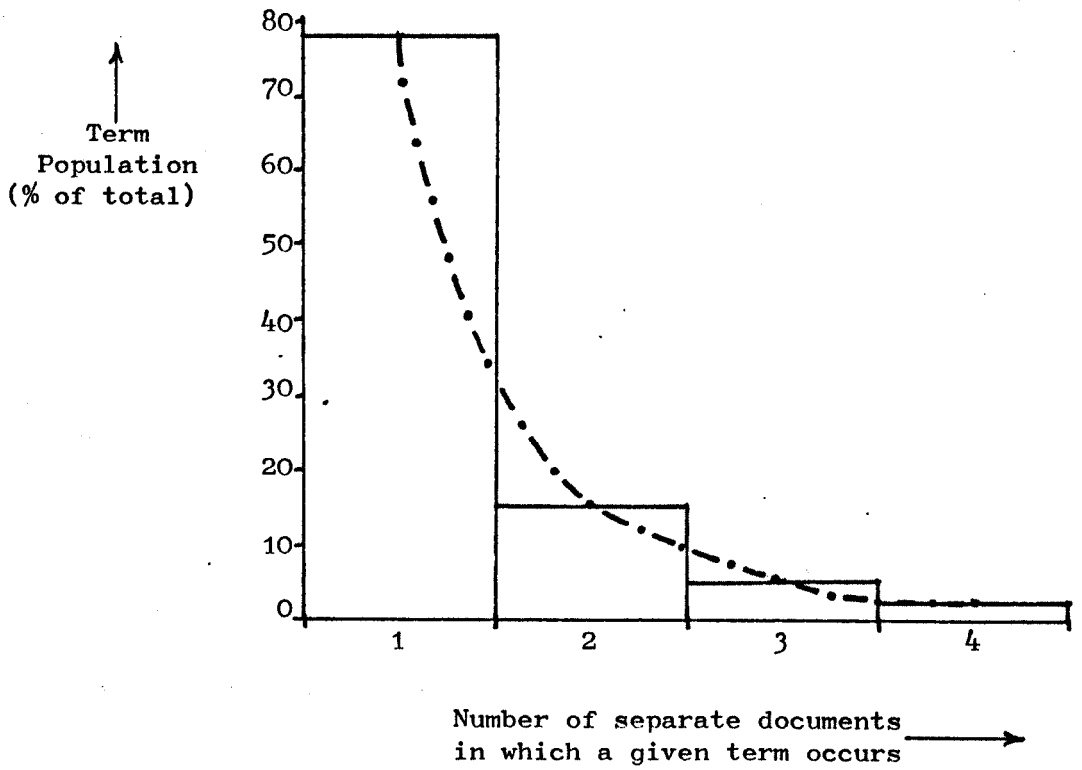


figure 7.2 - Term Population Distribution

This population histogram shows that most terms occur in only one document. Common words were removed before plotting this result, but if they were not filtered out, the effect would be to boost the high frequency end.

With synonym control, one would expect a general shift to the right to result, since different words are made equal when synonymously joined. In fact, this effect is hard to recognize when synonym control is applied to the whole stock of data, but is more interesting when one considers:

(a) a UDC class (documents 1-5, Car Crashes) and
(b) the equivalent computer clustered class (documents 1,3, & 4)

Figure 7.3 is a re-plot of figure 7.2, but the abscissa is now a percentage (to ensure normalization), and curves are used to promote clarity. Now however, the term distributions are plotted for individual classes, and at each of three levels of synonym control:-

(i) None
(ii) Low Level
(iii) High Level and Low Level

One should, ideally, see 9 curves, but synonym control does so little to effect the distributions for either (1) all the data taken together, or (2) and individual UDC class, that the 3 curves are practically superimposed in each of these cases. For the computer clustered class however, synonym control does produce the predicted left-right shift, and the reason for this is allied to the reason for the separation between UDC class and computer cluster on the abscissa, namely the total dependence of a clustering system on words (not concepts).

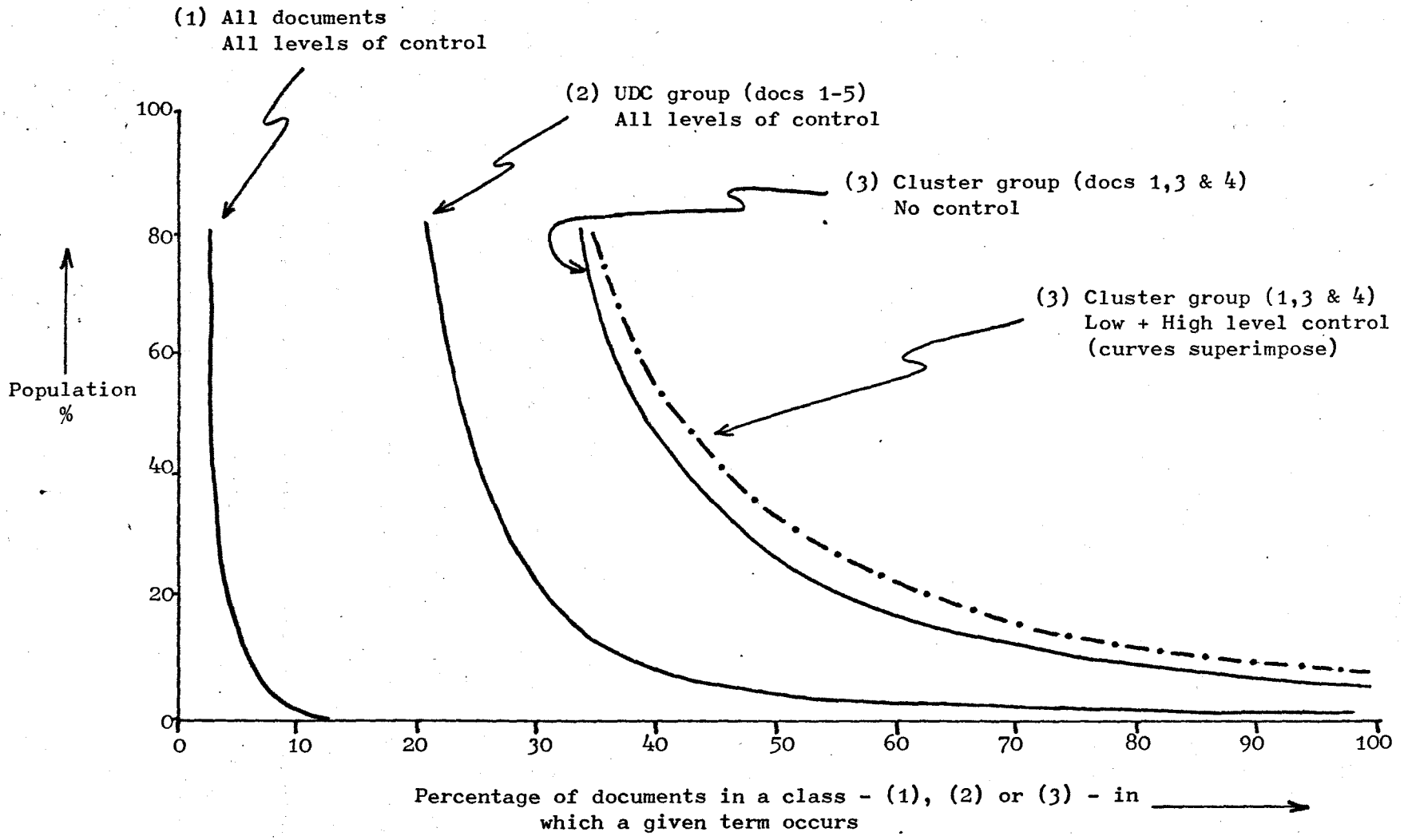


figure 7.3 - Term Distributions by subject class

Viewed statistically, the curves that are further towards the right represent groups of greater coherence, since individual words occur with greater frequency, thereby consolidating the cluster. But the UDC class, which by this reckoning exhibits poor coherence, has obviously been constructed by an intelligent classifier, so why the difference? The answer of course, is that the classifier has interpreted the card subject matter in a far more complex manner, drawing connections between phrases to which simple automatic word processing is blind. It is those yet more complex synonyms and word relationships that separate curves (2) and (3) in figure 7.3 and herein lies the gap which a successful clustering system must bridge.

If however, it is assumed that the UDC is at least a reasonable organization, and one that is not directly opposed to an ordering based on natural language prose alone, then the possibility exists for building a clustering system within a pre-existing UDC framework. The above experiments with pure clustering showed that minimal control (high objectivity) and sensible threshold adjustment, cause clustering to support rather than ignore UDC classes. If, therefore, UDC classes were to be used to collect vectors, rather than using on explicit clustering algorithm, then one could approach a system that would respond to prose to indicate relevant UDC areas. In other words, a system for the automatic attribution of UDC (numbers could be produced.)

Such a system would combine the automatic aspect of clustering, with the steadying influence of the UDC (notably synonym control). In a large stock of data, a given UDC class would have a large number of words associated with it. Synonyms would be dealt with by the brute force of quantity, and words of high information value would be those occurring with high frequency in a few discreet areas only, and not spread evenly over many classes.

A program was written that would accept prose and associated UDC numbers (manually attributed). For each piece of prose, a term vector was constructed and made to point to the attributed UDC numbers (one or more). In this way, vector sets were defined by common UDC attributes rather than by an intrinisic properties of the vectors themselves, so the sets were not exclusive (a vector could be placed in any number of classes.)

Unclassified prose was then submitted to the program, and formed into the shape of a vector. Comparison of this prose set vector with pre-existing vectors thus provided suggestions of UDC numbers likely to be applicable to the new piece of prose. For the comparison process, terms were weighted after the inverse of their frequency of occurrence (rare terms being assumed to be of greater "information value" - SALTON[126]). No external synonym control was exercised.

Before going into greater detail, I'll give a practical example of the programs attempt to assign UDC numbers to a piece of prose. First the computer was fed information on a range of subjects associated with Energy. It was then asked to assign likely UDC numbers to a piece of prose based on previous experience. Each line of output consists of a suggested UDC number, a measure of the degree of certainty with which this number was proffered (on an arbitrary scale), and a brief literal description of the UDC number:

```
ENTER DOC PROSE (#TERM, ## ALL TERM)
>ASSIGN
ENTER PROSE (#TERM)
>NUCLEAR POWER WORKERS OUTSIDE WINDSCALE PLANT PROTESTING
>AT PROPOSED CUTS IN ENERGY EXPANSION PROGAMME.
>#
621.039.577.007.25                0548  NUCLEAR POWER STATION WORKERS
621.039.577(425.6WINDSCALE)       0499  WINDSCALE
621.31.004.641                    0271 *ELECTRICITY GENERATION AND SUP
662.76.004.641                    0249  GAS CUTS
35.075.2:621.039.577(425.6WIND0249  WINDSCALE ENQUIRY
621.039                           0106 *NUCLEAR ENERGY
-89                               0098 *NUCLEAR POWER (AS MOTIVE FORCE
621.039.577                       0098 *NUCLEAR POWER STATIONS
621.039.542                       0055 *NUCLEAR FUEL
66(42)621.039                     0049  UK NUCLEAR POWER INDUSTRY
ENTER COMMAND
```

The program was tested using Subject Profiles taken from the Film Library Classification Department in the Energy subject field, to provide UDC/prose combinations with which to teach the computer connections. Each computer assignment was compared with a separate manual assignment based on the same profiles, and only the first ten computer suggestions were used in each case (see later). These comparisons are contained in figure 7.4.

Example 1

```
ENTER COMMAND
>ASSIGN
ENTER PROSE (#TERM)
>MEETING OF THE ENERGY ADVISORY COUNCIL HELD TO DISCUSS THE
>EFFECTS OF LATEST NUCLEAR TECHNOLOGY ON THE ELECTRICITY
>GENERATION AND SUPPLY INDUSTRY. SHOTS OF ELECTRICITY PYLONS
>AND HARWELL ESTABLISHMENT WITH VOICE OVER.
>#
621.315.66                          1123  ELECTRICITY PYLONS
727.5:621.039(421.9HARWELL)         1022  HARWELL ATOMIC ENERGY ESTABLIS
35.078.2EAC                         0772  EAC (ENERGY ADVISORY COUNCIL)
621.039                             0605 *NUCLEAR ENERGY
621.31.004.641                      0478 *ELECTRICITY GENERATION AND SUP
621.311                             0416 *ELECTRIC GRID
351.824.112:06.043GASCOUNCIL        0249  GAS COUNCIL
66.073                              0166  GAS SUPPLY
66(42)621.039                       0127  UK NUCLEAR POWER INDUSTRY
537                                 0124  ELECTRICITY
```

Suggested UDC nos.

Degree of confidence in the suggestion (arbitrary scale)

Example 2

```
ENTER COMMAND
>ASSIGN
ENTER PROSE (#TERM)
>PANORAMA FILM REPORT CONCERNING THE CRISIS IN THE COAL
>INDUSTRY. VARIOUS SHOTS OF COAL TIPS AND COAL DELIVERIES
>BEING MADE. MENTION OF COMPETITORS WITH VOICE OVER OF GAS
>RIGS, NUCLEAR POWER STATIONS AND OIL FIELDS.
>#
662.66.004.34                       1075  COAL DELIVERIES
622.799.7                           1075  COAL TIPS
621.039.577                         0862 *NUCLEAR POWER STATIONS
533.94                              0575  COAL FIELDS
622.32()                            0538  OIL FIELDS
66.073.2                            0533  GAS RIGS
620.91                              0499  ENERGY CRISIS
621.311.2                           0376  POWER STATIONS
622.242                             0288 *OIL PLATFORMS
62(42)621.31                        0242  POWER INDUSTRY
```

Example 3

```
ENTER COMMAND
>ASSIGN
ENTER PROSE (#TERM)
>FILM ITEM ON CALDER HALL NUCLEAR POWER STATION.
>SHOTS OF THE CONTROL ROOM AND STATION WORKERS ENGAGED IN
>VARIOUS ACTIVITIES. MENTION OF WINDSCALE ENQUIRY AND US
>PLUTONIUM BAN. SHOT OF A PARAFFIN STOVE HEATING THE
>SECURITY GUARD'S QUARTERS. BRIEF REVIEW OF UK NUCLEAR POWER
>POLICY
>#
621.039.577(425.6CALDERHALL)  1998  CALDER HALL
35.07(73).01:669.824.004.71   1997  US PLUTONIUM BAN
35.075.2:621.039.577(425.6WIND1498  WINDSCALE ENQUIRY
621.039.577.007.25            1430  NUCLEAR POWER STATION WORKERS
665.53                        0999  PARAFFIN
66(42)621.039                 0548  UK NUCLEAR POWER INDUSTRY
669.824                       0499  PLUTONIUM
621.039.577(425.6WINDSCALE)   0499  WINDSCALE
665.462                       0499 *DOMESTIC HEATING OIL
621.039.577-55                0465  NUCLEAR POWER STATION CONTROL R
```

Example 4

```
ENTER COMMAND
>ASSIGN
ENTER PROSE (#TERM)
>HORIZON SPECIAL ON ALTERNATIVE SOURCES OF ENERGY, WITH
>SPECIAL EMPHASIS ON SOLAR ENERGY AND GEOTHERMAL ENERGY.
>SHOTS OF SOLAR HOUSES. MENTION OF WAVE ENERGY SYSTEMS.
>COMPARISON WITH CONVENTIONAL SOURCES. SHOTS OF OIL LAMPS AND
>AND STEAM ENGINES.
>#
577.4:620.9                   1545  ENERGY SOURCES (ALTERNATIVE)
728:620.9:523.7               1198  SOLAR HOUSES
620.9:536                     1046  GEOTHERMAL ENERGY
620.9:532.59                  1046  WAVE ENERGY
683.8                         1038  OIL LAMPS
620.9                         0296 *ENERGY
620.9:523.7                   0246  SOLAR ENERGY
536.423.1                     0199  STEAM
-81                           0199 *STEAM AS MOTIVE POWER
621.1                         0099  STEAM TURBINES
```

Example 5

```
ENTER COMMAND
>ASSIGN
ENTER PROSE (#TERM)
>SCIENTIFIC DISCUSSION OF NUCLEAR POWER. COMPARISON OF
>THERMAL AND FUSION REACTORS. DISCUSSION OF FUELS SUCH AS
>PLUTONIUM AND URANIUM. TALK OF THE JET PROJECT. MENTION
>OF ANTI-NUCLEAR FEELINGS OVER SHOTS OF THE WINDSCALE PLANT.
>#
301.162.11:621.039            2053  ANTI-NUCLEAR FEELINGS
621.039.52                    1332  THERMAL REACTORS
621.039.6                     1332  FUSION REACTORS
621.039.6.001.5               0999  JET PROJECT (JOINT EUROPEAN TAU
669.824                       0499  PLUTONIUM
661.879.1                     0499  URANIUM
621.039.577(425.6WINDSCALE)   0499  WINDSCALE
662                           0499  FUELS
621.039.5                     0388  NUCLEAR REACTORS
[BNF]                         0277  BNF (BRITISH NUCLEAR FUELS)
```

Example 6

```
ENTER COMMAND
>ASSIGN
ENTER PROSE (#TERM)
>PROGRAMME ON THE UK ELECTRICITY SUPPLY SYSTEM. TALK OF
>THE CEGB WITH SHOTS OF ELECTRICITY PYLONS AND CABLES.
>ELECTRIC GRID DISCUSSED, AND THE USE OF ELECTRICITY FOR
>TRANSPORT. TALK OF ENERGY RECYCLING AND CLOSING SHOTS OF
>ELECTRICIANS.
>#
621.315.2                          1190  ELECTRIC CABLES
621.315.66                         1123  ELECTRICITY PYLONS
621.3.007.25                       0999  ELECTRICIANS
621.311                            0765 *ELECTRIC GRID
629.1-83                           0599  ELECTRIC POWER (TRANSPORT)
351.824.111.008.1CEGB              0561  CEGB (CENTRAL ELECTRICITY GENE
 BOARD)
620.9.004.86                       0546  ENERGY RECYCLING
66(42)621.039                      0499  UK NUCLEAR POWER INDUSTRY
621.039.7.004.86                   0249  RECYCLING OF NUCLEAR WASTE
621.31.004.641                     0229 *ELECTRICITY GENERATION AND SUP
```

Obviously, the computer can make no judgements beyond language, so in Example 3 for instance, the UDC number for "paraffin" is suggested, although this constitutes only an aside to the central theme. This language base provides some advantages however, such as in Example 4 where the more general concept of "steam power" is suggested, although only "steam engines" were explicitly mentioned in the prose. It was this form of happy accident (which is really due to the nature of language) that gave rise to the acceptable assignments in fig. 7.4.(Note! - Only certain of these additional assignments will be applicable. For instance, in Example 1, the "electric grid" number is suggested, but is not strictly relevant to the prose. As such, these additional suggestions can neither be viewed as failures or successes, but rather as embellishment).

Example Number	% of manual assignments matched by the program	% of remaining computer assignments that were acceptable
1	100	100
2	60	50
3	75	66
4	100	100
5	88	100
6	100	0
Average	87	70

figure 7.4 - Automatic UDC attribution performance

Despite these encouraging results, reliable performance assessments cannot be quoted for a pilot system bearing an unknown relation to any operational system that it might herald (tempting though this may be, given a measure of success). Factors that render extrapolation risky include:-

(i) Change of prose.

The more extensive prose contained on UDC catalogue cards would provide a richer source of initial data (i.e. UDC/word relations) than the profiles used above. Quantity of prose would offer compensation for the synonym control that I explicitly tried to avoid, but certain words would doubtless arise as interference that could not be filtered out in the same way that consistent common word noise can be removed.

(ii) Threshold settings

An operational system would need to take account of the scores used to assess suggestions (see previous examples) and be able to

rely upon a threshold value, below which results are not to be trusted. (rather than use the 'first-ten' strategy employed above). My own experiments were inconclusive in this regard - thresholds varying from 8% to 50% of the maximum score were acceptable in different cases, and so an average would be meaningless at such a high level of deviation. Without a cut-off point, the user would be required to make the necessary judgements. In a large scale system, empirical derivation of a threshold might, or might not, be possible.

(iii) Synonym control

Up until now, I have taken for granted the control of synonyms likely to exist in a large system. If a given class (UDC number) has several hundred cards (vectors) associated with it, most of the terms connected with the subject will be covered in all their varying grammatical forms. This is however, an assumption, since disruptive synonyms are bound to occur, thus providing another source of noise not so easily removed.

Through all this doubt however, one can discern elements of potential:-

(1) A system for automatically attributing UDC numbers would obviously be a great boon to the Classification section at the Film Library, and even if such a system were only one of suggestion, it would save on a good deal of spade work. It is of course possible to imagine problems, notably that supervising classifiers might be blinkered to seeing only what the computer suggests, thus failing to make their own, more subtle, assignments.

(2) The system learns UDC-prose connections from any source. Although when setting up the system initially, UDC catalogue

cards would offer the best primary source of data, there is no reason why defficient areas should not be "patched up" by someone sitting at a terminal and providing the program with data from any number of sources - her own head even. Experience and performance figures would indicate areas that needed tuning up in this way.

(3) If UDC numbers can be assigned to documents, they can also be assigned to queries (provided that the prose is not too abrupt). One therefore has the basis of a fully automatic natural language retrieval system based on an original UDC structure. In the same way that the LEARNER program has the capacity to grow apart from the UDC organization, so could this system build upon a UDC sub-structure (the original organization that it was "taught" by UDC). Eventually, perhaps even working in concert with LEARNER, one could imagine a natural language system in which UDC keys are viewed as little more than pointers to areas of mass storage, subject definitions being completely taken over by characteristic vectors.

This wouldn't be a matter of dragging the UDC toward automation, but rather harnessing the services of an existing, reliable information organization in stepping towards a fully automated, natural language retrieval system. Some years ago the UDC was suggested as a "switching language" for translation between classifications, but now it seems that this role might be extended to embrace less conventional counterparts.

Before moving on, I think a couple of flowcharts might be in order. The first, figure7·5, compares clustering with the UDC assignment program, whilst the second, figure7·6, depicts a natural language retrieval system based on automatic UDC assignment - see (3) above.

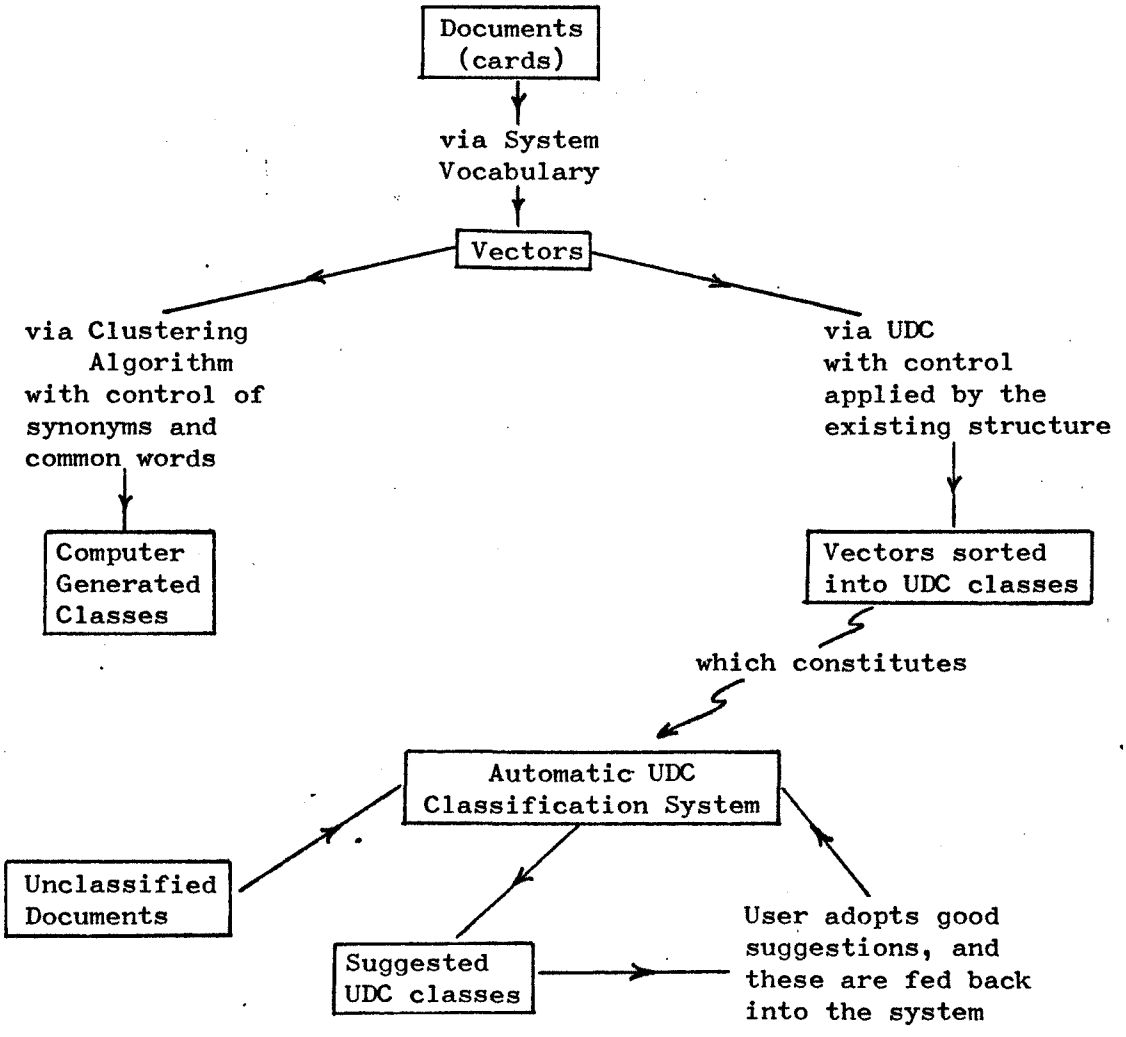


figure 7.5 - Automation in Classification

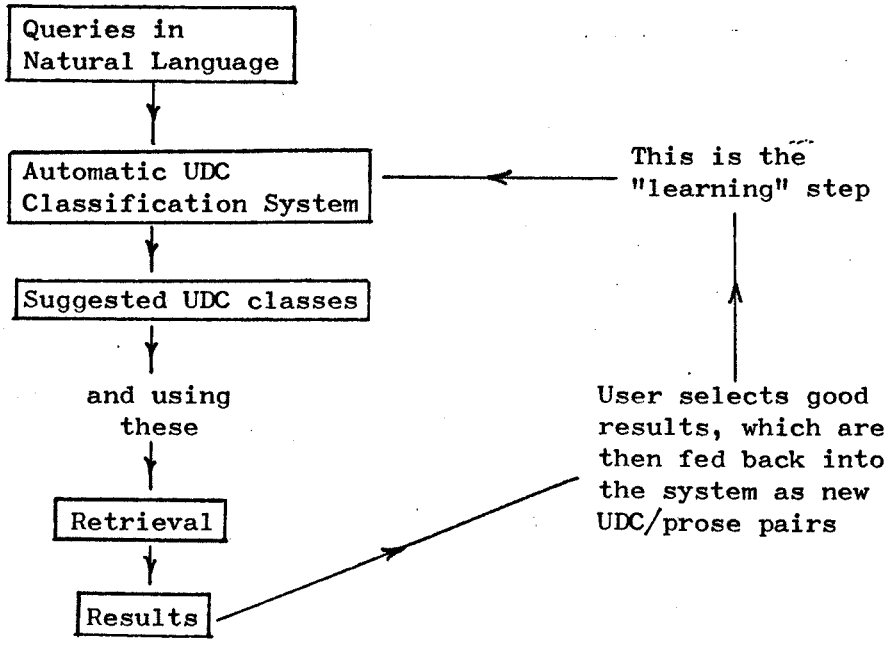


figure 7.6 - Learning from users in a natural language retrieval system

As an aside, I might say that it proved possible to simulate the automatic system described in figure 7.6 by using:-

(i) the automatic UDC assignment program, followed by

(ii) the LEARNER program (see prvious Chapter).

The assignment program was first fed a section of the UDC Authority File, and a short piece of prose was then submitted to it by way of a query. Retrieval based on the UDC entries thus suggested was subsequently performad using LEARNER.

In the example that follows, the middle section links the two programs, and would not be part of an operational system. The output is ranked in order of correlation to the original query (the second item does not satisfy the date constraint):-

```
>@XQT PILOT.CLASSER
ENTER COMMAND
>ASSIGN
ENTER PROSE (#TERM)
>CENTRAL HALL WESTMINSTER IN 1946
>#
(421.221CENTRALHALL)              0999  CENTRAL HALL, WESTMINSTER
"1946"                            0499  THE YEAR 1946 AD
ENTER COMMAND
>@
END OF RUN
>@XQT FILMX.LEARNER

ENTER COMMAND
>INPUT
ENTER TERM (# TERMINATOR)
>CENTRAL HALL, WESTMINSTER
>THE YEAR 1946 AD
>#
      CENTRAL HALL, WESTMINSTER
      (421.221 CENTRAL HALL)
>YES
      THE YEAR 1946 AD
      "1946"
>YES
ESTIMATED SEARCH TIME = 0000 SECONDS FOR SCORED SEARCH
ESTIMATED SEARCH TIME = 0000 SECONDS FOR EXACT SEARCH
000   1=0001 2=0002

ENTER COMMAND
>SEARCH

ENTER COMMAND
>DISPLAY
  SCORE=020  UDC= 341.123.042:06.091.2:725.8(421.221 CENTRAL HALL)
                  "1946"
>
  SCORE=010  UDC= 341.123.042:06.091.2:725.8(421.221 CENTRAL HALL)
>
```

Linkage Section Only

Clustering in an integrated retrieval system

So far in this chapter, I have only considered clustering as a means of making other things possible (such as automatic UDC number attribution) and the clustering experiments, described earlier in this chapter, were by no means conclusive enough to suggest that a lowly 50% success rating is all that one could hope for. Whilst on the subject of automatic document classification however, it is useful to speculate on how such techniques could be assimilated into the type of modular retrieval system first mentioned in Chapter 4, and to be considerably expanded upon in Chapter 8.

A completely clustered document collection presents a conceptually ordered information structure in the same way as the UDC or any other of the classical organizations. Although one off clustering proves a lengthy and expensive exercise, it has the advantage of offering a fixed framework than can be tailored for rapid retrieval (eg. clusters can be built into hierarchies). This also implies that updating can be performed rapidly, but (and this applies to any conceptual structure) extensive updating introduces the possibility that the original organization will be prone to a degradation brought about by adding-on new entries locally. In other words, only by frequent re-organization of the whole collection, can a balanced representation be maintained.

In a modular retrieval system however, one would often be in the position of having only an isolated area of the collection under examination (the rest of the collection having been excluded in preceding steps of the process) and so one could demand that such a region be clustered from scratch as part of the retrieval process. This "demand clustering" could, of course, only be applied to small subsets of the collection, but it would remove the need for a clustering run over the whole collection at any time, as well as the

worries about localized updating that were expressed above. Demand clustering would almost certainly prove less efficient retrieval-wise than a fully clustered system, since a large stock of data is obviously richer in the variation of terms upon which the inferential aspect of clustering is based. But it would doubtless provide the user with a further useful string to his bow and, what is more, one capable of presenting a particularly user-friendly aspect. This last point emerges from the experiment described below, in which the demand clusters are shown to the user as groups of commonly occurring terms, i.e. the terms occurring most frequently in the separate clusters. Also likely clustering times would be predicted in advance, so that the user could always be persuaded to try other methods to partition the collection (eg. UDC retrieval) prior to embarking upon demand clustering (see Chapter 4.)

The experiment

The data consisted of the 98 documents (UDC card prose) used for clustering. Common words were removed manually and simple synonyms were linked. These steps were felt to be justified since they could be easily incorporated in a mechanical sense by way of look-up tables.

Having clustered the data, the individual clusters were examined automatically, and terms contained within each cluster were ordered according to frequency of occurrence, thus yielding groups such as the following:-

Moors Murders

>
BRADY
>
CARS
>
CHARGE
>
COURT
>
CROWD
>
HINDLEY
>
IAN
>
LARGE
>
MEN
>
MOORS

Amin/Hills

>
AMIN
>
BRITAIN
>
CALLAGHAN
>
DEATH
>
DENIS
>
FOREIGN
>
HILLS
>
JAMES
>
SENTENCE
>
UGANDA
>
VISIT
>
ACTED
>

Black Panther

>
BLACK
>
COURT
>
DONALD
>
HEIRESS
>
LESLE
>
MURDER
>
NEILSON
>
PANTHER
>
RONALD
>
TRIAL
>
WHITTLE
>

Stonehouse Case

>
JOHN
>
STONEHOUSE
>
BAILEY
>
BUCKLEY
>
CHARGE
>
FORMER
>
OLD
>
SECRETARY
>
SHEILA
>
TRIAL
>
DAY
>
GUILTY
>
MP
>
ALLEGED
>
DAVID

Princess Anne kidnap attempt

>
ACCUSED
>
ANNE
>
ATTEMPT
>
BALL
>
IAN
>
KIDNAP
>
PRINCESS
>
REP
>
TRIAL
>
BAILEY

Herrema kidnap

>
CAR
>
COURT
>
COYLE
>
DR
>
DUBLIN
>
DUTCH
>
EDDIE
>
GALLAGHER
>
HERREMA
>
INDUSTRIALIS
>
KIDNAP
>
LAST
>
MARIAN

Thus the user was presented with a set of classes characterized by representative terms which, assuming a well-informed user, would be as valuable as a more extensive piece of prose. Also, because clusters formed at low thresholds are more general than those formed at higher thresholds, the possibility exists for refining classes by making the threshold tighter, eg:-

JOHN, STONEHOUSE, BAILEY ... etc. @ Threshold = 1

becomes

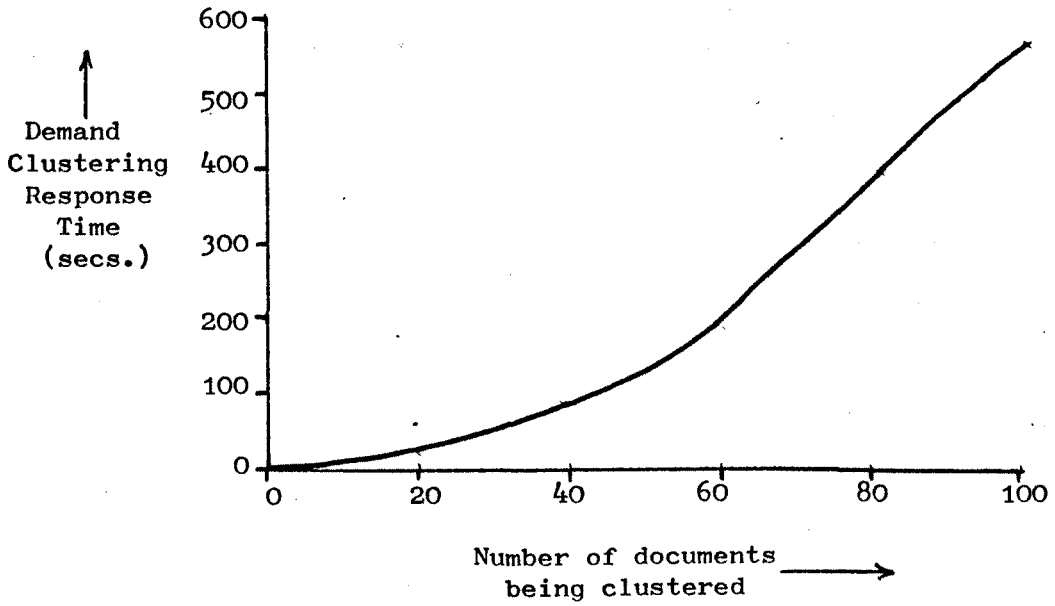
BAILEY, BUCKLEY, GUILTY ... etc. @ Threshold = 2

ie. by increasing the threshold setting, a particular aspect of the Stonehouse case (that concerning Sheila Buckley) is isolated

The problem with this approach, and it is one that haunts other aspects of clustering (eg. CROFT [247]), is that miscellaneous items that do not fit into clusters fail to be represented. One can place such items in a special (artificial) cluster to which the user is directed if no other cluster provides a satisfactory repository, but this "dustbin" cluster is of no conceptual virtue, and is simply a house-keeping device.

The speed of demand clustering was uninspiring - the construction of a similarity matrix being a somewhat protracted process. The actual response time plotted against the number of documents being clustered is shown in fig 7.7 and the expectation that clustering time would be influenced by the square of the number of documents being considered (because of the similarity matrix) is borne out. Clustering 98 documents took about 9 minutes response time (3minsCPU), but such figures should be taken lightly for an experimental system written in inefficient code (COBOL - which is most certainly unsuited to this type of computation, owing to its poor matrix handling capabilities.)

figure 7.7 - Effect of collection size on clustering time



This small experiment was conducted to illustrate one way of usefully incorporating clustering in a modular design - it was not intended as a definitive proof of viability. The characterization of clusters by commonly occurring terms was particularly well-received and there is no reason why this aspect should be limited to classes derived by cluster. Taking UDC groupings for example, such characteristic term lists could be derived for use at various stages (eg. schedule maintenance, UDC attribution etc.)

In a production system, the prime use of demand clustering would be to analyse sets of documents partitioned by other methods, but still as yet unmanageably large. For this to be practical, it would be necessary to analyse anything up to 200 documents in a minute or so - a performance criterion that I would consider within the bounds of present possibility if a low level language and more sophisticated clustering algorithms were used.

SUMMARY

This brief forray into the realm of document clustering was primarily intended as a bridge-building exercise between the UDC and its modern counterparts. Enough promise was found in clustering, and the several variations of practice to which it was applied, that one could justifiably expect further work to be fruitful. In this regard, one could also be assured that the stabilizing influence of the UDC would be an infallible protection against extensive system failures, without restricting the meritorious aspects of the more novel approach. Concertive system operation would again, therefore, appear to strongly reccommend itself.

What is more, the acquisition of computerized Film Library data, which is sure to be a gradual process, will suit the simultaneous development and enhancement of clustering applications. Also, if, as is likely, complete catalogue cards are stored on the computer, applications based on prose will not require any additional data input.

Extensive practical testing at the Film Library of the programs described in this chapter was not possible, owing to the librarian's workload. In short demonstrations however, enough enthusiasm was shown, particularly in the program for automatic UDC assignment, to suggest that aspects of clustering should be further researched as the BBC's trend towards computerization continues.

It is hard to suppress the conclusion that one stands to lose little, but to gain much, from some committment in this area.

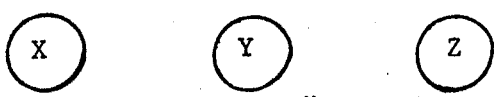
CHAPTER 8

A modular approach to retrieval system design

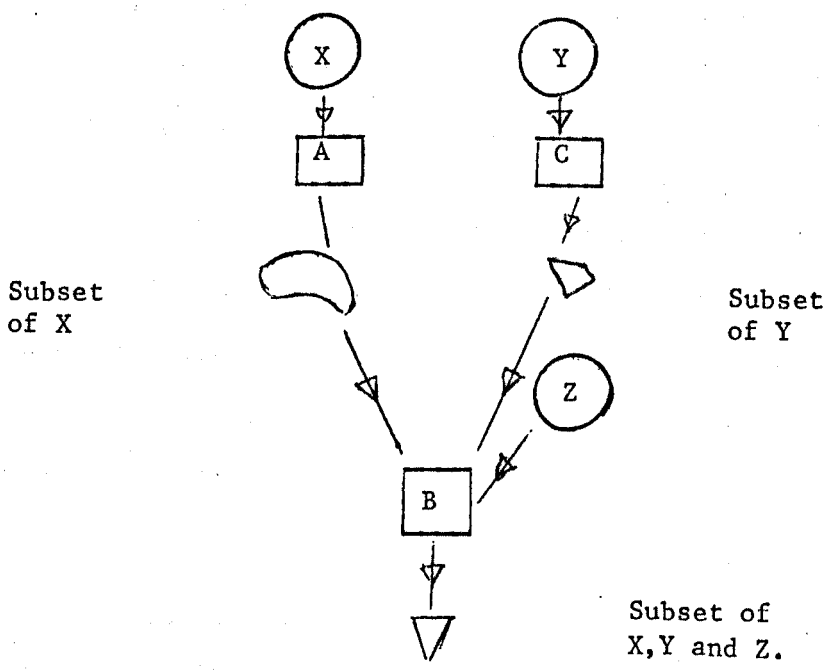
In Chapter 4, the view of information retrieval as a partitioning operation was introduced, and in Chapters 5, 6 and 7 methods comprising individual partioning functions were described. It was also shown, in Chapter 1, that enquiry types are commonly switched (subject to title for example) in order to rescue failed retrievals, and in Chapter 2 the interdependence of the various film library files was highlighted in the results of the statistical survey.

The theme common to all of the above points, is that combinations of diverse retrieval methods and diverse data files could usefully, or perhaps even vitally, be amalgamated into a modular retrieval system able to respond dynamically to queries. The LEARNER program is one approach to the provision of dynamic response to the expectations of the user, and flexible structuring of the retrieval system itself is another. Given a set of retrieval methods ("modules") :-

and a set of data files:-



it might be possible to construct an "ad hoc" retrieval system from a selection of these components, to best suit any particular (and unpredicted) query . For example:-



When the concept of "modularity" is extended to cover the data files as well as the retrieval procedures, the parallel with Relational Databases becomes obvious.

Ever since the pioneering work by CODD on the relational approach (refs. 248 to 250), the normalized tabular representation of data has attracted many purists who seek the completely independent structuring of data that the textbook definitions of bona fide databases stress. There are some rather esoteric arguments continuing (such as that invoked by those who demand that relations should never be more than binary), but generally speaking, the solid basis of the relational concepts cannot be muddled,

and the purity of relations is assumed when the "third normal form" is reached. (It should be noted however, that adding to a relation in third normal form can lead to redundancy, and this has caused a "fourth normal form" to be promulgated in those circumstances where this degradation can occur - DATE[255] (1977)).

A measure of the applicability of the relational model in the setting of the Film Library, can be seen in the following example where the separate files (relations) are used concertively to produce a specific result.

Given the following relations:-

UDC

U	A
621	X29
621	Y43
743	Y27

NAME

N	A
SMITH	X29
SMITH	Y27
WILLS	X42

U = UDC number A = Accession Number

Suppose it is required that all UDC numbers should be printed out that have a name reference in common with UDC number 621. This can be achieved in terms of operations performed on relations UDC and NAME, where intermediate results are placed in temporary relations T1, T2 etc, (used for work space only):-

(1) SELECT from UDC (U,A) such that U = 621 → T1 (U,A)

which produces: T1

U	A
621	X29

(2) JOIN T1 (U,A) with NAME (N,A) on domain A→ T2 (U,A,N)

which produces:

T2

U	A	N
621	X29	SMITH

(3) PROJECT T2 (U,A,N) on domain N→ T3 (N)

which produces:

T3

N
SMITH

(4) JOIN T3 (N) with NAME (N,A) on domain N→ T4 (N,A)

which produces:

T4

N	A
SMITH	X29
SMITH	Y27

(5) JOIN T4 (N,A) with UDC (U,A) on domain A→ T5 (U,A,N)

which produces:

T5

U	A	N
621	X29	SMITH
743	Y27	SMITH

(6) PROJECT T5 (U,A,N) on domain U→ T6 (U)

which produces:

T6

U
621
743

(7) SELECT from T6 (U) such that U ≠ 621→ T7 (U)

which produces:

T7

U
743

This stepwise process could have been expressed more succinctly, but recorded in this expanded form the mechanics of the relational operations are better displayed.

An important point to note, is that in a relational model, the concertive use of data elements (JOIN operation) provides report generating possibilities that individual partitionings cannot supply. It is this effective restructuring of the elementary data relations that provides such valuable possibilities for increasing the apparent information content of a database. Also, from the point of view of the Film Library, enquiry type definitions (shown to be something of an anomaly, in that they reflect user's misconceptions) are less restrictive, since the ability to combine relations has the effect of multiplying the access paths which can be explored in a single retrieval manoeuvre. But just as important (for my purposes at least) as this inherent characteristic of the relational approach, is the potential for modular design under the aegis of a relational database.

Any operation serving to partition a relation without affecting the column structure (ie. reducing the cardinality without affecting the domains) can, for present purposes, be regarded as a SELECT operation. Seen in this light, the RETRIEVE program, LEARNER, and a process of demand clustering (see Chapters 5,6 and 7) can be viewed as highly specific methods of SELECT partitioning where a relation is input to the retrieval method, and subsequently output by the retrieval method in, one hopes, a refined form.

Any retrieval device satisfying this simple relation-in relation-out formula can, therefore, be made an integral part of a relational database. Any such module is defined in terms of the fields (domains) upon which it operates. For example, if RM1 is a retrieval module that bases its selection upon a character field and an integer field, then it can be used to SELECT from any relation containing domains of these two types - it being up to the user to ensure that the operation

will be meaningful.

In such a system applied to the Film Library therefore, RETRIEVE, LEARNER and CLUSTER modules could be used in flexible combinations suited to any particular query requirement. For example, given the following basic relations:-

"UDC" UDC catalogue

U	A
621.4	X49
621.5	Z22
622	Y43

"TERM" List of terms on UDC cat. cards.

T	A
Apple	Z22
Boy	X12
Car	Y47

the following could be achieved in conjunction with the conventional relational operations:-

(1) Apply RETRIEVE to UDC(U,A) → T1(U,A)

(2) JOIN T1(U,A) with TERM(T,A) on T → T2(U,A,T)

(3) Apply CLUSTER to T2(U,A,T) → T3(U,A,T)

where RETRIEVE operates on the U domain of the UDC relation, and CLUSTER operates on the T domain of the TERM relation to give the final result.

In the same way that relations are registered with controlling relational software, so retrieval modules are "Plugged in". As new retrieval methods are needed, or, in the case of clustering, are proven viable, they can be added to the operational system and assimilated into existing practices for serving users. However, choice of retrieval method should not be the concern of the casual user, nor indeed, for most users at any rate, should even the most cursory acquaintance with the relational approach be a prerequisite of system operation. One therefore needs some method of mediation between a relationally oriented retrieval system and the novice enquirer.

No matter how elegant a retrieval system may be, one of the greatest tests of its practical applicability is the ease with which it can be used. To build into a system all those levels of interface that users can be expected to require is not generally a feasible suggestion, firstly because the user's ability is hard to predict, and secondly because software overheads can become burdensome. This being the case, the most obvious alternative is then to rely on exhaustive documentation and training, which is not usually desirable.

With this in mind, it was decided to investigate a method for the "ad hoc" building of retrieval system interfaces based on building blocks comprising of:-

(i) retrieval modules, and

(ii) data relations

A mediating information analyst aware, on the one hand,of user needs and, on the other hand, of the existence of such components within a relational framework, could then build interfaces, and indeed systems themselves, of the appropriate level of complexity. To do this however, it would be necessary to have a design language in which to express the juxtapositioning of the building blocks which the analyst arranges, and it was in this area that work was begun.

A detailed account of this aspect is contained in APPENDIX 3, which is a paper on the subject awaiting publication, and to which the reader is now referred.

The design language was primarily implemented to facilitate interface and retrieval system design based on the standard relational operations within procedures (MACROS) structured by the analyst. In the same way that relations can be added as the need arises, so can specific

retrieval modules be included as variants of the SELECT operation. If, for example, a new method of UDC retrieval were developed to serve a particular need, the algorithm could be made available for use by registering it with the interpreter behind the design language, and then referencing it in the design runstream.

Example:- A retrieval module called "PEOPLE" retrieves all UDC strings containing the .007 group that also contain specified UDC classes. It could be used to operate on the U domain in the UDC relation:-

UDC

U	A
621	X29
621.007	Z43
624.6	Y22
624.007	X26

So one could write:-

ANS(U,A) = PEOPLE [UDC(U=621,A)]

which would result in:-

ANS

U	A
621.007	Z43

This simple example could be placed in a runstream such as the following, where the same result is computed for classes 621 to 625:

```
1.    REL UDC(U/C(60), A/C(10))
2.    REL ANS(U/C(60), A/C(10))
3.    OPEN UDC
4.    UVAR = 620
5.    PLUSAB UVAR
6.    IF U > 625 THEN 10
```

```
7.      ANS(U,A) = PEOPLE [UDC(U=UVAR, A)]
8.      PRINT ANS
9.      GO TO 5
10.     END
```

Given a relation NAME:-

NAME

N	A
JONES	X24
SMITH	Z73
YOUNG	Y49

The results from the above example could be additionally qualified by a name reference, WILSON say:-

```
1.      REL UDC(U,A), NAME(N,A), ANS(U,A,N)
2.      OPEN UDC, NAME
3.      UVAR = 620
4.      PLUSAB UVAR
5.      IF UVAR > 625 THEN 9
6.      ANS(U,A,N) = PEOPLE [UDC(U=UVAR, A)]*(A!=A)*NAME(N="WILSON",A)
7.      PRINT ANS
8.      GO TO 4
9.      END
```

These analyst written retrieval procedures would be activated by the casual user as required, who would be asked to provide any necessary parameters. For example, the user interface might appear as follows (user responses are underlined):-

```
ENTER PROCEDURE NAME:- NAMØØ7
This procedures retrieves items that contain a supplied
UDC number in the same facet as the .007 group, and also
contain a supplied NAME reference.

ENTER UDC NUMBER:- 621
ENTER NAME:- WILSON, H

 A          U           N
X274      621.007     WILSON, H
Y293      621.43.007  WILSON, H
DONE
ENTER PROCEDURE NAME
```

The overall retrieval system would thus be structured around analyst designed procedure MACROS written with the users in mind, and at the necessary levels of complexity. In their turn, the procedures would be structured in terms of the available retrieval modules and relations. The relationship between the various people involved in the production and use of the database is shown in figure 8.1.

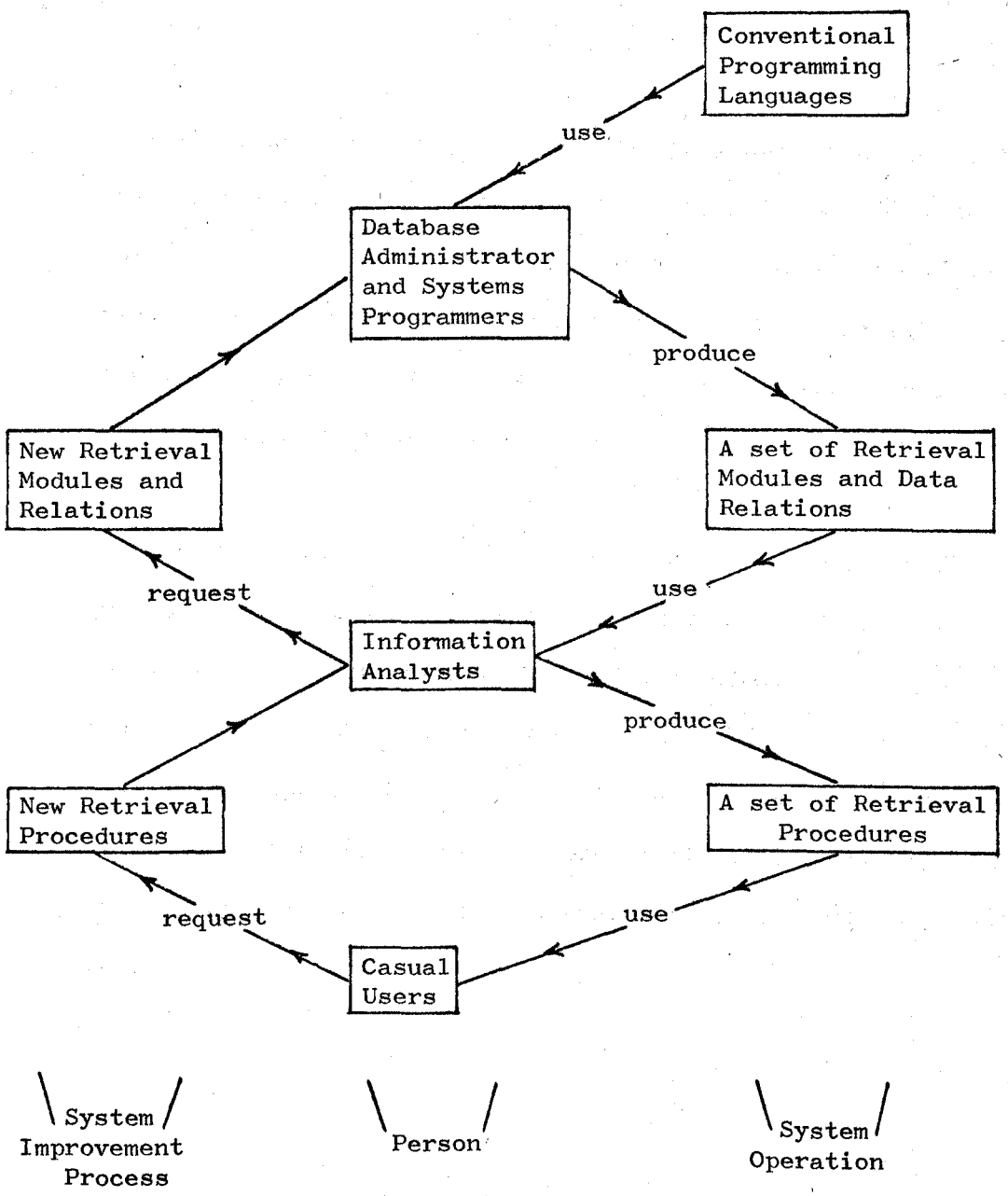


figure 8.1 - Overall structure of a modular system

Over the past few years, the BBC has been sponsoring research into the implementation of a relational database on its ICL computer - HUTT[251]. My recommendations that this approach should be exhaustively investigated in the context of the Film Library, would not therefore be alien to the BBC, as might be the case if some unknown commodity were being suggested.

The implementation of a relational database need not be of first priority, since any of the specific retrieval methods that I've discussed, and particularly RETRIEVE and UPDATE, would work perfectly well on their own. The supervising and binding aspect of the relational approach however, would be a boon to any specific system operating within it, and the theoretically solid data organization in a relational database (dependencies etc.) would be particularly desirable.

CHAPTER 9

Conclusions

The following points emerge from this work:-

(i) An efficient storage and retrieval system was developed which, being based on the UDC, was both reliable and readily available for adoption at the Film Library.

(ii) A modular approach to retrieval within the framework of a relational database was derived, and held to be of considerable applicability to this and many other retrieval systems.

(iii) A retrieval interface incorporating an ability for the system to learn from enquiries was examined, and felt to hold particular promise.

(iv) Document clustering was examined, and assessed to offer certain possibilities, the viability of which, in a system as large as that at the Film Library, would require a separate study by way of assessing the variables involved in scaling up.

A further major conclusion to be drawn, is the likelihood that these separate points might - under the aegis of point (ii) - be brought together as concertive methods of information retrieval, used collectively to cope with a wide range of query types. Another advantage of a modular approach is that space can be left for retrieval devices (eg. clustering) whose viability might be currently limited by hardware performance that, in time, one can justifiably expect to improve.

As regards hard and fast design suggestions, a number of specific points have already been agreed between myself and Film Library personnel. In the more general sphere of system design however, the following suggestions are offered:-

(1) That the individual files (both these currently used at the Film Library, and those proposed to open other retrieval paths - see Notes A & B below) should be accommodated as relations within a relational database.

(2) That specific methods of retrieval and relation SELECTION should be possible within individual relations, and notably:-

(3) That a UDC-based retrieval system, along the lines of the pilot RETRIEVE system, should be available for use within a UDC relation.

(4) That the system allow for the continued appraisal of the method for learning from enquiries - see note A below:

(5) That the system be made flexible enough to allow for the possible introduction of other methods at a later date - see Note B below.

Note A - LEARNER would feed off the existing files (relations) and also use another relation in which to store its own dynamically expanding vocabulary - a more free version of the Subject Index (which should remain inviolable).

Note B - Even from an early stage, document clustering experiments could be continued. Having a large amount of data available would make it easier both to reach valid conclusions, and to estimate with greater reliability the scaled-up viability of such methods. Special purpose relations would be needed to support clustering algorithms.

In preparing the literature survey that constitutes Chapter 3 of this work, the diversity of approach in Information Retrieval became, at times, almost painfully obvious. To be sure, it is only by covering a wide range of alternatives that the best applications can be reliably assessed, but the diversity of approach to storage and retrieval of information more probably reflects a serious division between the old and new schools - in which the latter currently lacks solidarity - rather than a genuine quest for objective, scientific completeness.

If there was any doubt beforehand, the work described herein has surely illustrated that a classical IR scheme - the UDC - can be subjected to any amount of sophistication by way of computerization. This is not a property of the UDC, but is an acknowledgement of the power of computers, and of the ability of modern software to solve the most intransigent problems of the data-handling variety.

Given a new tool - a computer - the urge to try new methods is almost irresistable, and various methods have sprung up over the last couple of decades that have evolved in this way, but then quickly faded as novelty alone proved to be an inadequate quality. This phase is now passing, and revolutionary computer applications have a genuine role to fill as the new masters of Information Science.

The various clustering experiments described in Chapter 7, which were based on a form of prose not geared to the computer, but geared to normal Film Library operation, were a refreshing indication that computers need not demand radical change (although such can easily be arranged if felt desirable) but can be used to stretch existing systems in whatever direction, and to whatever extent, the user demands, requires or simply finds interesting.

If only one point were to be appreciated from my work, I would ask that the reader comes to view a retrieval system as a flexible, multi-faceted entity, capable of being moulded (by the user or a mediating analyst) so as to update itself in terms of retrieval approach, with the same alacrity with which the data itself can be updated. The LEARNER program showed the way in which a single facet of the system could be made to exhibit a degree of hospitality, and the modular approach extended this concept to the totality of the retrieval system.

It is in combination of methods for the sake of the user, rather than by proof of a particular method for the sake of its author, that the greatest potential utility of a system lies. In responding to the efficacy of the various retrieval modules - adding a little sophistication here, removing a little extraneous effort there - the system itself can progress in time, so that a particular device becomes established on its merits, rather than on the number of appearances that it makes in the literature, although one hopes that the two would go in step.

Although it is the case that many of the points made in this thesis are specific to an extraordinary establishment, it is nevertheless true that they exemplify the reaction of the average library to the expanding option of computerization. Most librarians see, on the one hand, exciting "Tomorrows World" examples of the benefits of computerization but, on the other hand, a large existing system and a body of staff trained (and perhaps happy) in its exclusive operation. In order to weld these two halves together, the computer specialist must aim his attentions at the area of compromise between them, however intellectually soggy this ground might appear to be.

Seen in this light, the modular approach discussed in this thesis becomes less a possibility and more a necessity. The chance of using computerized conventional systems and experimental retrieval methods within a single framework - perhaps concertively - presents itself not only as a means of increasing retrieval performance, but also as a mechanism of breaching the generation gap between library retrieval methods.

REFERENCES

The following abbreviations are used:

J.ASIS.	Journal of the American Assn. for Information Science
LIB.J.	Library Journal
J.DOCN	Journal of Documentation
SP.LIB.	Special Library
ASLIB	ASLIB Proceedings
AM.DOC.	American Documentation
JACM.	Journal of the Association for Computing Machinery
CACM	Communications of the Assn. for Computing Machinery
ARIST	Annual Review of Information Science and Technology (pub. by ASIS and edited by Cuadra)
IS & R.	Information Storage and Retrieval
J.LIB.AUT.	Journal of Library Automation
ASIS	American Society for Information Science

1. COX, R.A. - Audiovis. Instr. 6:512-4(1961)
"Is your film library ready for automation?"

2. O'CONNOR, J. - J.ASIS. 24, 6:445(1973)
"Text searching retrieval of answer-sentences ..."

3. OTTEN, K. - J.ASIS. 21:89(1970)
"Towards a metascience of information: INFORMATOLOGY"

4. SHANNON, C. - Pub. by ILLINI (1949)
"The Mathematical Theory of Communication".

5. QUIGLEY, M.C. - LIB.J.66:1065-7(1941)
"Library facts from IBM cards".

6. FERRIS, L.A. et al - J.DOCN.3:250-71(1948)
"Bibliography on the uses of punched cards".

7. MAMALAKIS, M.J. - Wilson Lib.Bul. 16:350 (1942)
"More gadgets please".

8. QUIGLEY, M.C. - LIB.J. 58:318(1933)
"Guilds or technocracy".

9. COMPTON - Wilson LIB.BUL. 11:616-17 (1937)
"Bibliographic robot next?"

10. FAIR, E.M. - LIB.J. 61 (1936)
"Inventions and books - what of the future?"

11. KOHLSTEDT et al - LIB.J. 75: 417-18 (1950)
"Grand Rapids Public Library explores mechanical cataloguing".

12. HAND, W.J. et al - SP.LIB. 42:13-18 (1951)
"Special Library of the Future".

13. HARDKOPF,J.C. - LIB.J. 76:999-1001 (1951)
"Cybernetics and the library".

14. FAIRTHORNE, R.A. - J.DOCN.8:164-72 (1952)
"Automata and Information".

15. COMPTON - Dartmouth Col.Lib.Bul. 2:147-50 (1937)
"Less perspiration - more inspiration".

16. FRANCE - Int. Fed. Doc. Trans. 14: 113-15 (1938).

17. McCOY, R.F. - Wilson Bul.Lib. 14:399 (1940)
"Bookometer".

18. WÜSTER, E. - FID.R.DOC. 19:39-50 (1952)
"Visible and invisible alphabetisizing of UDC symbols".

19. RUSTON, W.R. - FID.R.DOC. 18, 1:31-40 (1951)
"A combined application of punched cards and the UDC for bibliography".

20. VAROSSIEAU, W.W. - FID.R.DOC. 15,2:41-6 (1948)
"Use of the UDC in selecting data with mechanical appliances".

21. SABEL, C.S. et al - J.DOCN. 18,3 (1962)
"Edge-punched card examination of retrieval patterns in information offices ..."

22. TAUBE, M. - AM.DOC. 6 (1955)
"Storage and retrieval of information by means of association of ideas".

23. BLACK, J.D. - ASLIB. 14,10 (1962)
"The keyword: in, abstracting, indexing and retrieving information".

24. JOLLEY, J.L. - ASLIB. 15 (1963)
"Mechanics of co-ordinate indexing and its relation to other indexing methods".

25. SCHULLER, J.A. - ASLIB. 12,11:372-89 (1960)
"Experience with indexing by UDC and Uniterms".

26. AM.DOC. 8:149 (1957)
"Concept co-ordination in information retrieval systems".

27. KENT, A. - ibid:149-51
"Evaluation of literature searching systems".

28. MYERS, J.M. - J.DOCN. 29,2 (1973)
"Computers and the searching of law texts in England and the USA ... "

29. PERRY, J.W. - AM.DOC. 5:18-22 (1954)
"Machine literature searching; a general approach"

30. HEUMAN, K.F. - SP.LIB. 51:483-4 (1960)
"Big black-box at your beck and call".

31. IVALL, T.E. - Philosophical Lib. (1960)
"Electronic computers: principles and applications".

32. WARHEIT, I.A. - SP.LIB. 51: 485-92 (1960)
"Principles of computer operation".

33. NOLAN, J.J. - AM.DOC. 10:27-35 (1959)
"Information storage and retrieval using a large random-access memory".

34. Control Engineering 8:22-4 (1961)
"Information retrieval: finding a needle in a haystack".

35. McCORMICK, E.M. - AM.DOC. 13:182-4 (1962)
"Trends in the use of computers for information processing".

36. DONLEY, A.M. - AM.DOC. 8:300-5 (1957)
"Library ecology and electronics".

37. LIPETZ, B. - "Information Storage and Retrieval" in"INFORMATION" pub. by FREEMAN for the SCIENTIFIC AMERICAN (1966).

38. BOURNE, C.P. - AM. DOC. 12:108-10 (1961)
"Historical development and present state-of-the-art of mechanized IR systems".

39. MOOERS, C.N. - AM.DOC. 11:229-36 (1960)
"Next 20 years in IR: some goals and predictions".

40. FREEMAN, R.R. - J.DOCN.20,3 (1964)
"Computers and classification systems".

41. SALTON, G. - JACM. 20,2 (1973)
"Recent studies in automatic text analysis and document retrieval".

42. SALTON, G. - Pub. by PRENTICE HALL (1971)
"The SMART Retrieval System".

43. SALTON, G. - CACM. 8 (1965)
"The SMART automatic document retrieval system."

44. CAMPEY, L.H. - ASLIB. Occsnl. Publcn. no.11 (1972)
"Generating and printing indexes by computer".

45. MARKUSON et al - Pub. by SYSTEM DEV.CORP (1972)
"Guidelines for Library Automation".

46. BECKER and HAYES - Pub. by WILEY (1970)
"Handbook of Data Processing for Libraries".

47. BARUCH, J. - ARIST. 1 (1966)
"Information system applications".

48. VERVLIET, H. - PROGRAM 8,3:117-33 (1974)
"Machine readable catalogues ...: an experiment with an interim MARC-compatible cataloguing system".

49. CHEN, S. - SP.LIB. 64:193 (1973)
"Automated cataloguing and reclassifications by ATS (Administrative Terminal System)".

50. ARIST. 1 (1966)
"New hardware developments".

51. DRACHMAN, A.G. - LIBRI 1,1:61-2 (1950)
"Microfilm rapid selector".

52. AVAKIAN, E.A. et al - SP.LIB. 48:145-8 (1957)
"AMFIS - the automatic microfilm information system ..."

53. LEWIS, C.M. - SP.LIB. 53:130-4 (1962)
"Interrelationship of microfilm, copying devices and IR".

54. SP.LIB. 55:582-3 (1964)
"Televised IR".

55. NEGUS, Reprographic Qtrly. 8,2 (1975)
"Automated microfilm display terminals - their role in on-line bibliographic IR systems".

56. LYNCH and SMITH - NATURE 230:153 (1974)
"Scientific information retrieval by computer".

57. LYNCH and SMITH - New Scientist 50:32 (1971)
"Information by the yard".

58. CORBETT - PROGRAM 4,4 (1970)
"Progress at AWRE in the development of mechanized current-awareness services".

59. BECKER, A.M. - AM.DOC. 19:311 (1968)
"Documentation and electronic data processing".

60. SCHNEIDER, J.H. - SCIENCE 173,3994 (1971)
"SDI and indexing of information".

61. MITCHELL, H.F. - AM.DOC. 4:16-17 (1953)
"Use of the Univac FAC-TRONIC system in the library reference field".

62. ERSKINE, P.L. - ASLIB. 15:41 (1963)
"Co-ordinate indexing - a bibliography".

63. MATTHEWS et al - ASLIB. 20,2 (1968)
"A case study of the UNITERM index".

64. LANE, B.B. - SP.LIB. 55:45-6 (1964)
"Keywords in - and out of - context".

65. VAN HALM, J. - SP.LIB. 63-10 (1972)
"Use of the UDC in a mechanized system: its applications in a KWIC program".

66. FREEMAN, R:R. - J.DOCN. 23,4 (1967)
"The management of a classification: modern approaches exemplified by the UDC project of the AIP".

67. SCHULLER, J.A. - ASLIB. 12,11 (1960)
"Experience with indexing and retrieving by UDC and Uniterms".

68. CLEVERDON et al - The CRANFIELD (ASLIB) Project.
(i) CRANFIELD I - (1962)
"Report on testing and analysis of an investigation into the comparative efficiency of indexing systems".
(ii) CRANFIELD II - (1966)
"Factors determining the performance of indexing systems" - 2 volumes.

69. SALTON, G. - SCIENCE 168 (1970)
"Automatic text analysis".

70. VICKERY, B.C. - J.DOCN. 16,4 (1960)
"Thesaurus - a new word in documentation".

71. REES, A.M. - AM.DOC. 13:93-4 (1962)
"Relevancy and pertinency in indexing".

72. SALTON, G. - JACM. 10:440-57 (1963)
"Associative document retrieval techniques using bibliographic information".

73. JACQUESSON and SCHIEBER - IS & R. 9:85-94 (1973)
"Term association analysis on a large file of bibliographic data, using a highly-controlled indexing vocabulary".

74. STILES, H.E. - JACM. 8:271-9 (1961)
"Association factor in IR".

75. SWANSON, D. - SCIENCE 132,3434 (1960)
"Searching natural language text by computer".

76. ELMAN, S. - SP.LIB. 66:12 (1975)
"Cost comparison of manual and on-line computerized literature searching".

77. HALL, NEGUS and DANCY - New Scientist 51:210 (1971)
"Towards instant information".

78. LANCASTER, F.W. and FAVEN, E.G. - Pub. by MELVILLE (1973)
"Information Retrieval On-Line".

79. BLACK, D. and FARLEY, E. - ARIST 1 (1966)
"Library automation".

80. MITCHELL, P.C. - JASIS. 24, 5:347 (1973)
"SOLAR: a storage and on-line automatic retrieval system".

81. MORROW, D.I. - JASIS. 27, 1 (1976)
"A generalized flowchart for the use of ORBIT and other on-line interactive bibliographic search systems".

82. CLOUGH, C. - J.DOCN 27,4 (1971)
"A single computer-based system for both current awareness and retrospective search: operating experience with ASSASIN".

83. BACK - JASIS. 23,2 (1972)
"... design of on-line retrieval systems".

84. BORMAN - ibid

"Interactive search of bibliographic databases".

85. WESSEL, A.E. - Pub. by MELVILLE (1975)

"Computer-Aided Information Retrieval".

86. DOYLE, L.B. - Pub. by MELVILLE (1975)

"Information Retrieval and Processing".

87. HENLEY, J.P. - Pub. by MACDONALD (1970)

"Computer-based Library and Information Systems".

88. MYERS, J.M. - J.DOCN. 29,2, (1973)

"Computers and the searching of law texts in England and the USA..."

89. ATHERTON and MILLER - J.LIB.AUT. 3,2 (1970)

"LC/MARC on MOLDS: an experiment in computer-based, interactive bibliographic storage, search, retrieval and processing".

90. VAN DYKE et al - J.LIB.AUT. 5,1 (1972)

"Multipurpose cataloguing and indexing system".

91. NEEDHAM and SPARCK-JONES - J.DOCN. 20 ,1 (1964)

"Keywords and clumps".

92. SPARCK-JONES, K. - J.DOCN. 26,2 (1970)

"Some thoughts on classification for retrieval".

93. SPARCK-JONES, K. - J.DOCN. 29,4 (1973)

"Automatic indexing".

94. WOLFF-TERROINE et al - J.DOCN. 27, 2 (1971)

"Computer-aided generation of a structured documentary language".

95. FIELD, B.J. - J.DOCN. 31,4 (1975)

"Towards automatic indexing: automatic assignment of controlled language indexing and classification from free indexing".

96. SHERA, J.H. - AM.DOC. 3:15-20 (1952)

"Effect of machine methods on the organization of knowledge".

97. BAKER, F.B. - JACM.9:512-21 (1962)

"IR based upon latent class analysis".

98. MARON and KUHNS - JACM. 7:216-44 (1960)
"On relevance, probabilistic indexing and IR".

99. TRITSCHLER, R.J. - AM.DOC. 15 :179-84 (1964)
"Effective information searching strategies without perfect indexing".

100. BECKER and HAYES - Pub. by WILEY (1963)
"Information Storage and Retrieval".

101. BOURNE, C.P. - Pub. by WILEY (1966)
"Methods of Information Handling".

102. JAHODA, G. - Pub. by WILEY (1970)
"Information Storage and Retrieval Systems for Individual Researchers".

103. MITCHELL, R. - Pub. by BINGLEY (1971)
"Information Science and Computer Basics".

104. VICKERY, B.C. - Pub. by BUTTERWORTHS (1973)
"Information Systems".

105. KOCHEN, M. - Pub. by MELVILLE (1974)
"Principles of Information Retrieval".

106. LAWLOR, R.C. - in "Advances in Computers", Pub. by ACADEMIC PRESS, vol.3 (1962)
"Information technology and the law".

107. WALSTON, C.E. - in "Advances in Computers", Pub. by ACADEMIC PRESS, vol.6 (1965)
"Information Retrieval".

108. JACKSON, D.M. - in "Advances in Computers", Pub. by ACADEMIC PRESS, vol. 11 (1971)
"Classification, Relevance, and IR".

109. CLEVERDON, C.W. - ASLIB. 22:538 (1970)
"Information and its retrieval".

110. BAR-HILLEL, Y. - AM.DOC. 8:103-13 (1957)
"Logicians reaction to recent theorizing on information search systems".

111. O'CONNOR, J. - JACM 11,4 (1964)
"Mechnical indexing methods and their testing".

112. GARDIN, J.C. - Unesco Bul.Lib. 14:2-5 (1960)
"Document analysis and information retrieval".

113. AUSTIN, D. - J.DOCN. 30,1 (1974)
"The development of PRECIS: a theoretical and technical history".

114. AUSTIN, D. - Pub. by the BNB Council (1974)
"PRECIS: a manual of concept analysis and subject indexing".

115. GUTHRIE et al - J.LIB.AUT. 5,2 (1972)
"Analysis of search key retrieval on a large bibliographic file".

116. COSTELLO, J.C. - AM.DOC. 13:414-19 (1963)
"Computer requirements for inverted co-ordinate indexes".

117. DAVIS, C.H. - SP.LIB.63:381 (1972)
"A simple program for weighted-term searching".

118. BOOKSTEIN, A. - SP.LIB.25,4:232 (1974)
"Hash coding with a non-unique search key".

119. ARIST. 1 (1966)
"File organisation and search techniques".

120. ANON
in "Advances in Computers", Pub. by ACADEMIC PRESS, vol.12 (1972)
"File organisation techniques".

121. KEEN, M. - ASLIB. 20,1 (1968)
"Search strategy evaluation in manual and automated systems".

122. MILLER, W.L. - J.DOCN. 27,4 (1971)
"A probabilistic search strategy for MEDLARS".

123. DYKE, H.G. - AM.DOC. 10: 85-6 (1959)
"Figure of merit ordering system for a search output".

124. ROBERTSON, S. - J.DOCN. 30,1 (1974)
"Specificity and weighted retrieval".

125. BELZER, J. - JASIS, 22:226 (1971)
"Justification for automatic indexing by frequency distributions of words".

126. SALTON, WONG and YANG - CACM. 18,11 (1975)
"A vector space model for automatic indexing".

127. SPARCK-JONES, K. - J.DOCN. 28,1 (1972)
"Statistical interpretation of term specificity and its application to retrieval".

128. SALTON, G. - JASIS. 23:11 (1972)
"The 'Generality' effect and retrieval evaluation for large collections".

129 YU, W.T. et al - JACM. 23,2 (1976)
"A statistical model for relevance feedback in I.R."

130. THOMPSON, D.A. - JASIS. 22:361 (1971)
"Interface design for interactive IR systems".

131. SPARCK-JONES et al - J.DOCN. 29,3 (1973)
"A test for the separation of relevant and non-relevant documents in experimental retrieval collections".

132. MEINCKE and ATHERTON - JASIS. 27,1 (1976)
"Knowledge space: a conceptual basis for the organization of knowledge".

133. LYNCH - IS & R. 9,4 (1973)
"Compression of bibliographic files using an adaption of run-length coding".

134. JOLLIFFE, J. - J.DOCN. 24,3 (1968)
"The tactics of converting a catalogue to machine readable form".

135. MOGHDAM, D. - JASIS. 26,3 (1975)
"User training for on-line IR systems".

136. MEADOW, C.T. - Pub. by WILEY (1967)
"Analysis of information systems; a programmers introduction to IR".

137. DOLBY, J.L. - J.DOCN. 27,2 (1971)
"Programming languages in mechanized documentation".

138. CUADRA, C. - JASIS. 22:107 (1971)
"On-line systems: promise and pitfalls".

139. HINES, T.C. - J.DOCN. 23,3 (1967)
"Computer manipulation of classification notations".

140. MØGAARD-HANSEN, R. - Tidskrift fut Dokumentation 24;1 (1968)
"UDC,DDC and LC in competition in the domain of the university library".

141. CLEVERDON, C.W. - ASLIB.12,12 (1960)
"ASLIB Cranfield research project on the comparative efficiency of indexing systems".

142. TELL, B.V. - 32nd Annual Meeting of ASIS. Vol.6 of Proceedings (1969)
"Document representation and indexer consistency:... using UDC and keywords".

143. MILLS, J. - ASLIB. 16:48 (1964)
"IR: a revolt against conventional systems?"

144. WAHLIN, E. - J.DOCN. 19,4 (1963)
"Principles for a universal system of classification ..."

145. MAROSI, A. - J.DOCN. 25, 3:185 (1969)
"Euratom thesaurus and the UDC".

146. ARNOLD, D.V. - J.DOCN. 14:183-9 (1958)
"IR; a pragmatic approach".

147. PERREAULT, J.M. - LIBRI 18,2:120-4 (1968)
"Automatized retranslatability of UDC codes".

148. AYRES, F. et al - J.DOCN. 23,1 (1967)
"Some applications of mechnaization in a large special library".

149. SAVILLE, J.P. - ASLIB. 16:283 (1964)
"The UDC in metallurgical literature classification: work at the Iron and Steel Institute".

150. PATTEN, M.N. - ASLIB. 26:189 (1974)
"Experience with an in-house mechanized information system".

151. NEVILLE et al - ASLIB. 27,5 (1975)
"BLIPS (Building Research Establishment Library Processing System) ..."

152. MAROSI, A. - J.DOCN. 25,3 (1969)
"Euratom thesaurus and the UDC - combined use for the subject organization of a small information service".

153. RUSSELL, M. and FREEMAN, R.R. - AIP/UDC Project, Report No.4 New York (1967)
"Computer aided indexing of a scientific journal by the UDC ..."

154. LESLIE, W.H. - ASLIB. 13 :145 (1961)
"Automatic retrieval of technical information".

155. GOLD, J.A. - Canadian Lib. J. 29:6 (1972)
"PRECIS: an analysis".

156. DAHLBERG, I. - J.DOCN.27,1 (1971)
"Possibilities for a new UDC".

157. CALESS and KIRK - J. DOCN. 23,3 (1967)
"An application of the UDC to machine searching".

158. McCASH, W. and CARMICHAEL, J. - J.DOCN. 26,4 (1970)
"UDC user profiles as developed for a computer based SDI service in the iron and steel industry".

159. ATHERTON, P. and FREEMAN, R. - AIP/UDC Project, Report No.5, New York (1967)
"File organization and search strategy using the UDC in mechanized retrieval systems".

160. de REGT, W.F. - Proc. Copenhagen Conference on the UDC in a mechanized retrieval system (1968)
"The use of UDC as an indexing language in computer-based information retrieval systems".

161. ditto no. 159.

162. BRANDHORST and ECKERT - ARIST. 7 (1972)
"Document retrieval and dissemination systems".

163. LLYOD, G.A. - ASLIB. 21:204 (1969)
"The UDC in its international aspects".

164. SWETS, J. - SCIENCE. 141,3577 (1963)
"IR systems".

165. BROOKES, B.C. - J.DOCN. 24,1 (1968)
"The measures of IR effectiveness proposed by SWETS".

166. HEINE, M.H. - JASIS. 25,3:183 (1974)
"Design equations for retrieval systems based on the SWETS model".

167. CLEVERDON and MILLS - ASLIB. 15 (1963)
"The testing of index language devices".

168. CLEVERDON, C.W. - ASLIB. 19:173 (1967)
"The Cranfield tests on index language devices".

169. SWANSON, D. - Lib.Qtrly. 35:1-20 (1965)
"The evidence underlying Cranfield results".

170. KYLE, B. - J.DOCN. 20,2 (1964)
"IR and subject indexing: Cranfield and after".

171. CLEVERDON, C.W. - SP.LIB. 55:86-91 (1964)
"Uncovering some facts of life in IR".

172. CLEVERDON, C.W. - J.DOCN. 28,3 (1972)
"On the inverse relationship of recall and precision".

173. CLEVERDON, C.W. - J.DOCN. 26,1 (1970)
"Evaluation tests of IR systems".

174. REES, A.M. - ASLIB. 18:316 (1966)
"The relevance of relevance to the testing and evaluation of document retrieval systems".

175. BOURNE, C.P. - ARIST. 1 (1966)
"Evaluation of indexing systems".

176. KREVITT et al - Information 6,2 (1973)
"Evaluation of information systems - a bibliography 1967-1972".

177. FAIRTHORNE, R.A. - J.DOCN. 21:267-70 (1965)
"Some basic comments on retrieval testing".

178. KENT, A. et al - AM.DOC. 6:93-101 (1955)
"Machine literatire searching; operational criteria for designing IR systems".

179. KATTER, R.V. - ARIST 4 (1969)
"Design and evaluation of information systems".

180. ROBERTSON, S.E. - J.DOCN. 25, (1969)
"Parametric description of retrieval tests".

181. TAUBE, M. - AM.DOC. 16,2 (1965)
'The psuedo-maths of relevance".

182. O'HARA, F.M. - JASIS 21:166 (1970)
"The corners and edges of the Precision-Recall square".

183. FARRADANE - J.DOCN. 30,2 (1974)
"Evaluation of IR systems".

184. AMICK, D.J. - JASIS. 21:171 (1970)
"A multivariate statistical analysis of the ude of a scientific-based, computer-aided IR system".

185. ROBERTSON, S. - J.DOCN. 30,3 (1974)
"A statistical analysis of retrieval tests; a Bayesian approach".

186. SARACEVIC, T. et al - J.DOCN. 23,1 (1967)
"Towards the identification and control of variables in IR experimentation".

187. ROBERTSON, S.E. - JASIS. 26,4 (1975)
"Explicit and implicit variables in IR systems".

188. SPARCK-JONES, K. - J.DOCN. 31,4 (1975)
"A performance yardstick for test collections".

189. COOPER, W.S. - JASIS. 24,2:87 (1973)
"On selecting a measure of retrieval effectiveness".

190. CUADRA and KATTER - J.DOCN. 23,4 (1967)
"Opening the black-box of Relevance".

191. SARACEVIC, T. - JASIS. 26 (1975)
"Relevance: ..."

192. SALTON, G. - AM. DOC. 16:209-22 (1965)
"Evaluation of automatic retrieval procedures: selected test results using SMART".

193. LANCASTER, F.W. - J.DOCN. 24,1 (1968)
"Evaluating the economic efficiency of a document retrieval system".

194. MILLER, W.L. - J.DOCN. 27,2 (1971)
"The extension of users' literatire awareness as a measure of retrieval performance, and its application ot MEDLARS".

195. HERSEY, D.F. et al - J.DOCN. 27,3 (1971)
"Free text word retrieval and indexing: performance profiles and costs".

196. DAMMERS. H.F. - J.DOCN. 31,1 (1975)
"The economics of computer-based information systems - a review".

197. LANCASTER, F.W. - JASIS. 22:12 (1971)
"A cost-effectiveness analysis of IR and dissemination systems".

198. VICKERS, P. - J.DOCN. 29,3 (1973)
"A cost survey of mechanized information systems".

199. MARTYN, J. - ASLIB. 21 :317 (1969)
"Evaluation of information handling systems".

200. KATZER - IS & R. 9,6 (1973)
"Cost performance of an on-line free-text retrieval system".

201. STEVENS, N:D. - AM.DOC. 12:234-6 (1961)
"Compartive study of 3 systems of IR".

202. KEITH, N.R. - JASIS, 21:237 (1970)
"A general evaluation model for an information storage and retrieval system".

203. SALTON, G. - JASIS 23:75 (1972)
"A new comparison between conventional indexing (MEDLARS) and automatic text processing (SMART)".

204. BORNSTEIN, H. - AM.DOC. 12:254-9(1961)
"Paradigm for a retrieval effectiveness experiment".

205. KLEMPNER, J. - AM.DOC. 13,3 (1964)
"Methodology for the comparative analysis of information storage and retrieval systems".

206. KATZER, J. - JASIS. 23:122 (1972)
"The development of a semantic differential to assess users' attitudes toward an on-line interactive reference retrieval system".

207. MARTYN and VICKERY - J. DOCN. 26,3 (1970)
"The complexity of the modelling of information systems".

208. HARLEY, A.J. et al - J.DOCN. 28,3 (1972)
"Towards a universal search formulation language for IR".

209. Dorking Conference - J.DOCN. 13:152-3 (1957)
"International study conference on classification for IR".

210. SP.LIB. 50:404 (1959)

"Standards for a common language conference".

211. SHERA, J.H. - LIB.J. 84:3386 (1959)

"Standardizing language for machine searching".

212. McCARN - IS & R. 7,6 (1971)

"Networks, with emphasis on planning an on-line bibliographic access system".

213. KNOX, W.T. - SCIENCE 133:1274-5 (1961)

"Effective use of information science".

214. DAVIS, R. - ARIST 1 (1966)

"Man-machine communication".

215. BORKO, H. - Pub. by WILEY (1967)

"Automated language processing".

216. CLIMENSON, W.D. et al - AM.DOC. 12:17883 (1961)

"Automatic syntax analysis in machine indexing and abstracting".

217. ELIAS, P. - JACM. 22,3 (1975)

"The complexity of simple IR problems".

218. THOMAS, P.A. et al - J.DOCN. 31, 1 (1975)

"A computer simulation model of library operations".

219. GRIFFITHS, J. - J.DOCN. 31,3 (1975)

"Index term input to IR systems".

220. WORTHEN, D.B. - J.DOCN. 31,1 (1975)

"Applications of Bradford's law to monographs".

221. BROOKES, B.C. - NATURE 224 (1969)

"Bradford's law and the bibliography of science".

222. MAYES, P. - J.DOCN. 31,4 (1975)

"The use of the Bradford-Zipf distribution to estimate efficiency values for a journal circulation system".

223. CARAS, G.J. - AM.DOC. 19:120-2 (1968)

"Computer simulation of a small information system".

224. FREEMAN, R. and ATHERTON, P. - AIP/UDC Project Report Nos.6-9, American Institute of Physics, New York (1968)

(i) "Evaluation of the retrieval of metallurgical document references using the UDC in a computer-based system" (No.6)

(ii) "AUDACIOUS - an experiment with an on-line, interactive reference retrieval system using the UDC as the index language in the field of nuclear science" (No.7)

225. WILLIAMS, M. - ARIST. 9 (1974)
"Use of machine readable databases".

226. QUIGLEY, M.C. - LIB.J. 77-1152-7 (1952)
"Ten years of IBM".

227. PERRY, J. - AM.DOC. 1:133-9 (1950)
"Information analysis for machine searching".

228. SALTON, G. - JACM. 20,2 (1973)
"Recent studies in automatic text analysis".

229. BATTEN, W.E. - ASLIB. 19:351 (1967)
"Communication: a teach-in for computer software specialists and documentationalists".

230. SAMUELSON, K. - ARIST. 6 (1971)
"International information transfer and network communication".

231. TAGLIACOZZO, R. - JASIS. 26,5 (1975)
"The consumers ...: a survey of utilization of MEDLINE".

232. COOPER, W.S. - JASIS. 21:385 (1970)
"On deriving design equations for IR systems".

233. LISTON, D.M. - JASIS. 22:115 (1971)
"A systems approach to the design of information systems".

234. British Standard 1000A:1961 (F.I.D. No. 289)
"Universal Decimal Classification" (1961)

235. FOSKETT, A.C. pub. by Bingley, London (1969)
"The subject approach to information"

236. PERREAULT, J. pub. by Bingley, London (1969)
"Introduction to the UDC"

237. MILLS, J. pub. by Rutgers (1964)
"The Universal Decimal Classification"

238. BAYER, R. & McCREIGHT, E. Acta Informatica, 1 (1972)
"Organization of large ordered indexes"

239. KNUTH, D. pub. by Addison Wesley, ps 473-479 (1973)
"The art of computer programming. Vol 3: Sorting & Searching"

240. HELD, G. & STONEBRAKER, M. Comm. ACM, 21, 2 (1978)
"B-trees re-examined"

241. BAYER, R. & SCHKOLNICK, M. Acta Informatica 9 (1977)
"concurrency of operations on B-trees"

242. McCREIGHT, E. Comm. ACM, 20, 9 (1977)
"Pagination of B^*-trees with variable length records"

243. SAMADI, B. Inf. Proc. Letters, 5, 4, (1976)
"B-trees in a system with multiple users"

247. CROFT, B. J.ASIS p.341 (1977)
"Clustering large files of documents using the single link method"

248. CODD, E.F. Comm. ACM, 13, 6 (1970)
"A relational model of data for large shared data banks"

249. CODD, E.F. Courant Comp. Sci. Symposia 6, Data Base Systems, pub. by Prentice-Hall, New york (1971)

"Relational completeness of data sub-languages"

250. CODD, E.F. Proc. 1971 ACM Sigfidet Workshop on Data Description, Access and Control, San Diego (1971)

"Normalized Data Base structure: a brief tutorial"

251. HUTT, A. PhD. Thesis, University of Southampton (1976)

"A relational database management system"

252. SALTON, G. pub. by Prentice-Hall (1975)

"Dynamic information and library processing"

253. STAFF, B. Unpublished internal report (1976)

"Results of a survey carried out in the Enquiries Department at the BBC Film Library during May 1976"

254. WEDEKIND, H. in Database Management, Klimbie & Koffeman (eds) pub. by North Holland (1974)

"On the selection of access paths in a database system"

255. DATE, C.J. pub. by Addison-Wesley (1977)

"An Introduction to Database Systems - 2nd Edition"

A P P E N D I X 1

(I apologise for the confusion, but this Appendix itself has four Appendices, lettered A, B, C & D)

USERS GUIDE TO THE COMPUTERIZED BBC FILM LIBRARY INFORMATION RETRIEVAL SYSTEM

Version 2 - December 1977

CONTENTS

INTRODUCTION

I'm loath to state any guidelines on how you should read this document, since the approach will vary from person to person, but generally speaking the first time user is best advised to read it through (including the Appendices), although I would suggest - assuming you have access to the terminal - that you try things out on the computer as soon as you can, preferably alongside someone already experienced in its use.

The examples in the text and in Appendix D are copied exactly from computer printout. In these examples, lines input by the user are characterized by having a '>' character in the first character position, whereas all lines printed by the computer lack this initial symbol. The examples contained in Appendix D may well prove to be more useful than the wordy explanations contained in other sections (personally, I have always found learning by example to be most effective) and so I would recommend that you try following these examples through as you become conversant with the system. You should note however, that some of the things done in the examples (which I hope are self-explanatory) are not useful in themselves, it's just that I've tried to illustrate the workings of the system, and this can be done by abuses as much as by uses.

The more experienced user will be able to look upon the Appendices as a source of reference (notably Appendix C) and so they are high on fact but low on explanation.

SECTION 1 - Contacting the Computer

The UNIVAC computer, on which the Retrieval System is operating, belongs to the Open University and is sited at Milton Keynes. The terminal at the Film Library is connected to the computer by the public telephone system, which limits its speed of operation (to around 30 characters per second).

Details of making the connection are to be found in Appendix A.

SECTION 2 - Using the Computer

As well as being calculating machines on a gigantic scale, computers possess means for storing masses of data to which they have extremely fast access. All of the information referenced by the Film Library Retrieval System is stored on magnectic discs which are frantically spinning round all the time the computer is working. The computer gets at the information stored on these discs by means of pickup heads.

The computer itself is a massive electronic organization that, however impressive it might be to look at, is of no practical use without someone telling it what to do. The computer is given the required operating instructions in terms of 'computer programs', and the Retrieval System that you will be using is one such program. Now, on a large computer like the UNIVAC at Milton Keynes, several people are running programs at the same time, so someone, or something, must make sure that these separate programs don't get in each others way, and must ensure that we don't do anything that we're not allowed to do (e.g. erase other people's data, break the machine etc.) This "something" is itself a computer program, and it's running all the time the machine is operating. Because of its supervisory capacity, this special program is called the "EXECUTIVE" by UNIVAC (other manufacturers have other names) and not only does the Executive (or EXEC) keep a fatherly eye on what we're doing, but it also requires that everything we do on the computer, we do via the EXEC.

As well as being our master, the EXEC also serves us. For example, if we were in sole charge of the computer, then getting information from the magnetic discs would involve us in an awful lot of work, but in fact all we need do is to make a short request to the EXEC, and the information we require is made available to us.

When you are operating the Film Library Retrieval System, it is the Retrieval System program with which you are conversing, and the EXEC lets you get on with it; the commands used in the Retrieval System are not understood by the EXEC. If something goes seriously wrong however, the EXEC may intervene, but it is more likely that the program will deal with the error and continue to operate normally.

Things will go wrong from time to time (they always do) and mostly they will be of little significance, but it might so happen that the Retrieval System program will get confused and the EXEC will step in to stop the execution of the program - this is no cause for concern (although it might worry me, since it means that my program is malfunctioning). It is also possible that the EXEC itself will get confused, and when this happens the machine will invariably break/down; but you can rest assured that it is up to the EXEC to guard against all the contingencies, so even if a breakdown is directly attributable to you (and this is extremely unlikely to happen) it is still the fault of the EXEC. In short, nothing you can do (save taking a hammer to the terminal) will damage the computer - it being the computers responsibility to protect itself from users. A breakdown is generally signified by the message:

FE MSG UNIVAC DOWN

being printed, and when this happens you should hang up the phone and try again later.

Starting work on the computer

Having connected the terminal to the computer (see Appendix A), you must notify it (via the EXEC) that you're starting a 'run'. All information passed to the EXEC is prefixed by the '@' symbol, which is known as the 'master space character' or the 'at' sign. This symbol, appearing as it does at the beginning of the line, serves to inform the computer

that the line is addressed to the EXEC. The computer tells you that it is expecting input by typing the '>' symbol, known as the 'prompt' or 'solicitation' character, and, having signed on, you reply to the first prompt symbol by typing:

@RUN␣RØSTFF,SCS/BS,BSTAFF

where Ø = the number zero

␣ = space

The whole line then appears thus:

>@RUN RØSTFF,SCS/BS,BSTAFF

Now as it stands, this line only exists at the terminal, but simply by pressing the 'RETURN' key on the terminal, the line is sent to the computer which, thanks to the EXEC, is informed that a "Run" is being commenced by a user who it will refer to as "ROSTFF"; that the cost of the run will be recouped from account "SCS/BS" (an O.U.Research Budget - in fact no money is involved, it is simply for administrative purposes); and that the information to be accessed during the run is that belonging to "BSTAFF" (me).

When this RUN message has been sent, the computer should reply by typing the date and time. If any other response is made you should refer to Appendix B.

The prompt character '>' always means that the computer is expecting you to type something. You have all the time in the world to reply (don't feel rushed) although you will be cut off after about 15 minutes if you haven't done anything - this is simply to protect users from leaving a

terminal connected to the computer by mistake (NB. The phone line is still connected when the computer breaks down or cuts you off automatically, and it can only be disconnected by hanging up.)

Generally speaking, you should only send input following a prompt (i.e. when the computer is ready for you), but there are circumstances when a prompt doesn't appear and you should send your message without it (e.g., when a line has been deleted or when an '@@X' type of command is used to abnormally terminate program execution - see later). If you send a command when the computer is not ready for it, you'll receive the somewhat curt message:

WAIT-LAST INPUT IGNORED

CORRECTING INPUT

If your typing is as bad as mine, you'll probably need to correct quite a lot of lines before sending them to the computer (by pressing the 'RETURN' key). If the line is badly wrong you can erase it all and try again by holding the CONTROL key down and pressing the X key (no carriage return is necessary), which will cause the terminal to jump to the next line for you to try again (NB. The '>' prompt character will not be printed in this case - you just type and send the new line without it.)

To delete a single character, hold down 'CONTROL' and at the same time press the Z key. Nothing will appear to have happened at the terminal, but the last character will have been deleted. To delete two or more characters, hold down CONTROL and press the Z key two or more times.

Examples: (i) >@RXNYP↑

CONTROL and X keys pressed here

@RUN

sent line reads: @RUN

(ii) >@RUGN
↑

CONTROL and Z keys pressed here

sent line reads: @RUN

(iii) >@RUGHIN
↑

CONTROL and Z keys pressed 3 times here

sent line reads: @RUN

Getting at the Retrieval System

Like the information it accesses, the Retrieval program is kept (when it is not in use) on a magnetic disc. In order to find the program, type:

@ADD␣FILMLIB.GO (␣ = space)

This has the effect of preparing the retrieval system for operation, as well as collecting the seven information files which contain the data used by the system (UDC numbers, Subject Index entries etc.). Typing the above statement should result in the following response:

7 "READY" messages

1 "FURPUR" message

6 "BLOCKS COPIED" messages

6 "READY" messages

If the computers response differs from the above, you should refer to Appendix B (see section "After typing the @ADD command"). If all goes well however, it means that everything is ready for you (the data files are "assigned to your RUN") and you can start the retrieveal system by typing:

@XQT␣FILMLIB.RETRIEVE (␣ = space)

where 'XQT' means "EXECUTE" (i.e. start going) the Retrieval program. In fact what happens is that the computer takes a copy of the Retrieval program (called "RETRIEVE") from the magnetic disc, and sets it going under the control of the EXEC, in the electronic heart of the computer. The 7 information files are left on the disc, but the Retrieval program can read from them when necessary.

Stopping the Retrieval System

(See also: Appendix B - "Unexpected Messages at any time".)

In the next section I go onto describe in detail how to use the Retrieval System, but before doing this it is vitally important to tell you how to stop the program and get back into communication with the EXEC. Firstly I should point out that no matter how you stop the Retrieval program - e.g. by blowing up the terminal, cutting the phone wires, or by one of the more orthodox methods descibed below - it is impossible to corrupt either the program or the information files. Recommended methods of stopping the program are as follows:

(1) NORMAL TERMINATION: Typing a single '@' in reply to the prompt character at any stage, will result in the program terminating normally with an 'END OF EXECUTION' message.

(2) ABNORMAL INTERUPTION (A): If the terminal is producing reems of output and it doesn't look like stopping in the near future, you should press the 'BREAK' key. The computer will reply (eventually) with an '*OUTPUT INTERRUPT*' message, and you should then type:

@@XØO (no prompt) O = the letter

This will result (again eventually) in the printing of the prompt character, to show that excess output has been discarded. You can either continue with retrieval or terminate - see (1).

(3) ABNORMAL TERMINATION (B): If you have issued a command and the computer has failed to reply after a long wait (i.e. it has not printed a '>', AND repeatedly pressing RETURN results in the *WAIT LAST INPUT IGNORED* message) then you can terminate the program by typing:

@@XØT (no prompt)

This will result in an '*EXECUTION TERMINATED*' message. If, in addition, a 'MORE INFO>' message is printed, you should press carriage return.

If you terminate by (1) or (3) above, you are now back in the hands of the EXEC program, and you can either:

(i) Sign off and hang up the phone, or

(ii) Re-execute the Retrieval program by typing:

@XQT FILMLIB.RETRIEVE

NB. There is no need to re-assign the files (by the @ADD statement) since they remain available until you have signed off.

When you have no more work to do for the time being, you should sign off. To do this type:

@FIN

which means "Finish". The computer will then print out several lines of accounting and timing information which, unless you are interested, is not important. When this is completed you will receive a '*TERMINAL INACTIVE*' message, and you should reply to the last prompt by typing:

@@TERM

which means "terminate the connection to Milton Keynes". Having pressed the return key for the last time (in sending the @@TERM command) you should hang up the phone.

EXEC commands (those beginning with an '@' symbol) can be issued at any time, assuming, of course, that the computer is ready to receive input. So at any point after the @RUN command, an @FIN will cause run termination (as opposed to a single '@' which causes Retrieval System termination).

NB. Hanging up the phone in mid-run is not to be recommended as a method of termination. It may, or may not,have the desired results.

Using the Retrieval System

The computer Retrieval System that is initiated by the '@XQT FILMLIB.RETRIEVE' command is, in many ways, directly analogous to the present manual system. Of the 7 files that have to be present (assigned to the run), five are used to hold computer-based versions of the Authority File, the Subject Index and the main UDC file. The discrepancy (5 computer = 3 manual) is due to changes in structure that are necessary to increase speed of retrieval, but, as far as you are concerned, the Retrieval System will seem to deal in terms of the 3 files with which you are conversant. The

sixth computer file is used to hold information generated during the operation of the Retrieval System, and it is discarded when you sign off. The seventh file contains UDC strings split up into their separate facets, thus permitting a novel method of retrieval. The relevance of these last two files will be exposed in due course.

Limitations

The program has been written to examine the feasibility of designing a system of retrieval entirely reliant upon the UDC and, as such, accession numbers, cataloguing information etc, has not been included - it would have cost a lot of money to have it put on the computer. The most likely mode of operation is for a user to start off with a subject and end up with a set of UDC strings (retrieved from the computer-based UDC file) that satisfy the user's requirements. I am, therefore, assuming perfect attribution of UDC numbers, in which case studying the cataloguer's prose is purely a formality. This might seem a bit idealistic, but for the present I want to test retrieval and UDC string manipulation, for which the present stock of information is adequate.

Talking of information, there now follow details of the UDC strings held in the computer-based UDC file:-

UDC area	Number of Strings
34-341.358.007	3348
[AEI] to [LADBROOKES]	561
(480) to (491.1 WESTMAN)	403
(=20) to (=943.5)	453
M74: 623.451 to M99"414.22"	460
R13(256) to R13(421)	339
	Total = 5564

The computer-based Authority file and Subject Index are sufficient to cover all of these areas except the companies (for which I haven't yet transcribed the relevant Authority file entries).

RETRIEVAL

Having sent the @XQT command, the Retrieval program responds with:

ENTER COMMAND

You are then able to enter any of the Retrieval commands (only the high levels ones in fact, but more of this in a moment). A full list of commands and their resulting action is given in Appendix C, but for the present I shall run through the more important Retrieval commands, to give you a feel for the system. In all examples, user input is characterized by the initial '>' character.

(1) TERM command

Example:

```
>@XQT FILMLIB.RETRIEVE

ENTER COMMAND
>TERM
ENTER TERM
>PARKER
   *REF:- 0199  PARKER COMMISSION ON NORTHERN IRELAND INTERNMEN
                341.345 (411) 06.045 PARKER
```

Action: The computer solicits a term from the user and searches the Subject Index for the nearest alphabetical match.

(2) EXPAND command (following on from TERM)

Example:

```
>EXPAND
   *REF:- 0013  "OZ" TRIAL
                34.096: 343.541
   *REF:- 0324  PALESTINIANS
                (=922)
   *REF:- 0199  PARKER COMMISSION ON NORTHERN IRELAND INTERNMENT
                341.345 (411) 06.045 PARKER
   *REF:- 0018  PATHOLOGISTS, FORENSIC
                340.67.007
   *REF:- 0111  PEARCE'S COMMISSION ON RHODESIA
                341.231 U01 (679) 3..0.7
```

Action: Alphabetically expand (±2) on a term in the Subject Index (here the term "Parker ..."). This is particularly useful for finding alternate spellings.

(3) Browse commands

First a word of explanation: "Browse commands" is a generic term to denote a set of commands that enable the user to browse through any of the three main files (Subject Index, Authority file and UDC file). Browse commands are only meaningful when you are directly referencing one of these files, whereas all other commands can be used at any time. For this reason, Browse commands are also known as 'low level commands', whilst all other commands (eg. TERM and EXPAND) have no limitations on their use (provided they are used correctly) and are known as 'high level commands'. When the 'ENTER COMMAND' message is printed, you must reply with a high level command.

I shall now run through the use of the low level commands:

Example:

```
>TERM
ENTER TERM
>MAYAGUEZ
    *REF:- 0086   "MAYAGUEZ" AFFAIR
                   341.225.1 (596)
>N 2
    *REF:- 0122   MILITARY AID (*)
                   341.232.1
    *REF:- 0273   MONS KLINDT
                   (489.5 MONS KLINDT)
>N-2
    *REF:- 0122   MILITARY AID (*)
                   341.232.1
    *REF:- 0086   "MAYAGUEZ" AFFAIR
                   341.225.1 (596)
>J 24
    *REF:- 0146   ORGANIZATION FOR EUROPEAN ECONOMIC CO-OPERATION
                   341.232.3 O.E.E.C.
>BASE
    *REF:- 0086   "MAYAGUEZ" AFFAIR
                   341.225.1 (596)
```

'N'Action: 'N' is a browse command. Browse commands can be issued after TERM or EXPAND commands since they directly reference a file. 'N' stands for "NEXT", i.e. print the next X items (Subject Index items when following TERM or EXPAND commands) where 'X' is a one or two digit number (1 to 99) separated by a space (␣) or a '+' sign from the 'N'. The alternative to N␣ (or N+) is N- (where '-' is the minus sign); for example, N-2 rather than N 2 (or N+2) causes the two <u>previous</u> Subject Index items to be printed.

You could equally well type N 24, N+69, N-43 etc., depending on how much of the Subject Index you want to see. 'N' on its own (not followed by a number) assumes N+1 and just the next item is printed.

'J'Action: J means "Jump". J 24 tells the system to jump forward 23 Subject Index items and print the 24th. As was the case with 'N', J␣24=J+24. To jump backwards you could J-8, J-73 or whatever. The jump parameter can be anything between 1 and 99.

'Base'Action: Return to the place in the file to which you first referred. No matter how much browsing you have done in the file, the 'BASE' command (or just 'B' if you like) will put you back to your point of entry (probably at the point resulting from a TERM search).

(4) RETAIN command

Example:

```
>TERM
ENTER TERM
>COD WAR
   *REF:- 0083  COD WAR WITH ICELAND
                341.225.1 (491.1)
>RETAIN 0083
SYSTEM STRING 01 IS 341.225.1 (491.1)

ENTER COMMAND
```

Action: Retain details of a given subject, specified in terms of the four digit 'reference number' (*REF) which must be four digits long. I shall have more to say about where these details are stored later; for now I'm just considering the RETAIN command to be a bridge between the TERM selection and the:

(5) SEARCH command (following on from RETAIN)

Example:

```
>SEARCH
0186 DIRECT HITS
0086 ROOT HITS

ENTER COMMAND
```

Action: Search the UDC file using the chosen term (here the term most recently subject to a RETAIN command). 'DIRECT HITS' means the number of strings on the UDC file that exactly match the UDC number comprising the RETAIN search string. 'ROOT HITS' means the number of strings that match the UDC number RETAIN'd as far as it goes, but are longer than the RETAIN'd UDC string, eg: 434,435,43(44) would all qualify as ROOT HITS of the RETAIN string 43.

To actually look at the UDC numbers referred to by a SEARCH, you must use the:

(6) BROWSE command (following on from SEARCH)

(NB. BROWSE in this case is an actual command, and is not simply a description to cover J,N and BASE low level commands.)

Example:

```
>BROWSE
COMMENCE AT FIRST ROOT HIT:- 0091/12
  SCORE=100  UDC= 341.225.1(491.1)061.3
>J-2
  SCORE=100  UDC= 341.225.1(491.1)
>N 5
  SCORE=100  UDC= 341.225.1(491.1)
  SCORE=100  UDC= 341.225.1(491.1)061.3
  SCORE=100  UDC= 341.225.1(491.1)061.3(42)
  SCORE=100  UDC= 341.225.1(491.1)061.3(42)
  SCORE=100  UDC= 341.225.1(491.1)061.3(491.1 REYKJAVIK)
```

Action: Permit browsing in the UDC file, starting from the first root hit (or last direct hit +1). The 'SCORE' is intended as a percentage measure of similarity between the search string (defined by RETAIN) and the UDC file entry being displayed. For all direct and root hits the SCORE will be 100%, but one is able to move about the UDC file by means of the J,N and BASE commands, and always the UDC numbers displayed are scored against the RETAIN string. The set of so-called "browse commands" can therefore be used following the actual high level 'BROWSE' command, but in this case they are applied to the UDC file (following TERM and EXPAND they apply to the Subject Index).

Interim Note

The commands mentioned so far enable you to make straightforward searches in a manner analogous to that used in the manual system. In some respects the commands that follow are of less importance, but I hope it will become apparent that these commands help to introduce new facilities of UDC retrieval, and therefore improve the efficiency of the basic system.

(7) AFILE commands

These commands (there are several of them, all starting with the word 'AFILE') are concerned with the Authority file.

Links in the Authority File run both crosswise (eg. 456, 457, 458) and vertically or hierarchically (eg. 456, 45,4). All the AFILE commands except 'AFILE SEARCH' need a starting point -- i.e. an Authority file entry - and this is provided by using the same'reference number' (*REF) as was used by RETAIN.

'Across' Example:

```
>AFILE 0083 ACROSS
   *REF:- 0083  COD WAR WITH ICELAND
                 341.225.1 (491.1)
>N-2
   *REF:- 0084  ICELAND COD WAR
                 341.225.1 (491.1)
   *REF:- 0082  FISHING REGULATIONS
                 341.225.1
>J 5
   *REF:- 0088  VIOLATION OF AIR SPACE
                 341.226
```

Action: 'Across' initiates crosswise browsing in the Authority file. The AFILE command, directly referring to a file as it does, can then be followed by browse commands (J, N and BASE) so that limitless browsing through the Authority file is possible.

'UP' Example:

```
>AFILE 0199 UP
   *REF:- 0199   PARKER COMMISSION ON NORTHERN IRELAND INTERNME
                  341.345 (411) 06.045 PARKER
UDC SUPERIOR
   *REF:- 0191   INTERNMENT
                  341.345
>
UDC SUPERIOR
   *REF:- 0189   PRISONER OF WAR CAMPS
                  341.34
>
UDC SUPERIOR
   *REF:- 0176   WAR, LAW OF
                  341.3
>TERM
ENTER TERM
```

Action: Shorten the specified UDC Authority string (from the right) and search for matches which may be hierarchically superior, i.e. work up the UDC schedule hierarchy. Pressing carriage return in answer to the prompt character causes the process to continue, whereas typing another high or low level command (TERM in the example) stops the progression. When the UDC string can be shortened no further, an 'EMPTY STRING and RETURN' message is printed, after which, low level commands are not accepted.

It is also possible to work <u>down</u> a hierarchy, but as this is slightly more complicated I shall leave explanation of it until later (Appendix C).

'AFILE SEARCH' Example:

```
>AFILE SEARCH
ENTER UDC STRING
>341
   *REF:- 0019  INTERNATIONAL LAW
                 341
>N-2
   *REF:- 0017  FORENSIC PATHOLOGISTS
                 340.67.007
   *REF:- 0018  PATHOLOGISTS, FORENSIC
                 340.67.007
```

Action: Find the nearest match in the Authority file to a UDC string entered by the user. You can then either browse around using the low level commands, or enter a high level command.

(8) MYUDC command

Example:

```
>MYUDC
ENTER UDC STRING
>(485 STOCKHOLM)

ENTER COMMAND
>RETAIN
USER STRING 02 IS (485 STOCKHOLM)
```

Action: Accept a UDC string from the user, possibly to be used for a subsequent SEARCH. Again (as was the case with TERM) details concerning the string are not retained until the RETAIN command is issued, but you should note that since no reference number can be applied to a UDC number that the user has dreamt up (as opposed to one existing on the Subject Index) then no further qualification is necessary (or possible).

Therefore, typing simply 'RETAIN' takes the UDC string entered following the preceding MYUDC command, and makes it eligible for a SEARCH.

(9) EDIT command (following on from MYUDC)

Example:

```
ENTER COMMAND
>EDIT
-(485 STOCKHOLM)
>    9 AARHUS)###
(489 AARHUS)

ENTER COMMAND
>RETAIN
USER STRING 03 IS (489 AARHUS)

ENTER COMMAND
```

Action: Correct a UDC number, character by character such that:

¥ = leave this character position as it is (on the line above).

= replace this character position by a space.

X = replace this character position by the character X

where X is any valid UDC character

This command will commonly be used to correct or alter a string input via MYUDC. When followed by the RETAIN command the edited string will be saved for further use. In fact, EDIT can be used on any occasion (not necessarily following a MYUDC entry); for example, an EDIT following a 'RETAIN 1234' will refer to the UDC string that is the subject of the RETAIN (*REF:-1234) - this is treated further in Appendix C.

EVERYTHING!

The time has come to discuss the more complicated commands, so as to inform you of all the essential features of the Retrieval System. Things might seem a little difficult on occasions, but in allaying any fears you might have, I would like to make two points:

(i) I am of the opinion that computer systems should be the tool of the people using them, and if a system is more complex than it needs to be then it is for the computer to make things easier, and not for the user to expend more effort.

(ii) I have, of course, tried to make the system as easy to use as possible, and by the use of default rules (i.e. the computer makes sensible assumptions and saves you supplying unnecessary details) I would hope that an initial mastery of the system should not be too hard to achieve.

In fact, I hope that the diversity of the system will appear to increase with the greater understanding that you attain - causing it to be used in a simple or complex fashion, depending upon your experience of it. The important point however, is that I expect the system to be amended in line with your requirements (this is why you are being asked to use it) so that it can be made easier to use and also, perhaps, more versatile. In this connection, I must stress the need for you to make any suggestions that you can, as regards facility of use or more general improvements. Doubtless some suggestions will be impossible to implement, but don't let that stop you.

Keeping Notes

Before going on to detail the commands themselves, I'll have to discuss what the computer does in terms of "keeping notes" on what you do in the course of a single program execution, since it is in the use of these stocks of "notes" that the real capabilities of the system become manifest. In fact, some of the commands already discussed have a note taking effect. For example, RETAIN causes details of a search string to be retained, and SEARCH causes the results of a search to be kept.

There are three places where these notes are kept for you; one of these is the sixth file that is assigned (see earlier), but since this has a very specific use, I shall avoid mentioning it for the moment. The other two places are known as the 'DICT' (Dictionary) and the 'PAD', and they can be defined as follows:

(1) The DICT (Dictionary) records details of all major transactions - a "major transaction" being one to which later reference is likely to be made.

(2) The PAD records details of searches (and some other things that we'll come to shortly) resulting from transactions recorded on the DICT.

It can be seen from these two definitions that PAD entries cannot be present without corresponding DICT entries; DICT entires can, however, be present without corresponding PAD entries. RETAIN for example, leads to an entry being made on the DICT, but no corresponding entry is made on the PAD until a SEARCH has been performed using the term that was the subject of the RETAIN.

When such a search has been performed, the DICT entry is supplied with a pointer to the relevant entry (or rather entries) on the PAD.

The DICT and PAD together keep a record of every major manoevre carried out during a single program execution, and when you're accustomed to their formats you'll be able to read them like a log book. Details of browsing, AFILE activities etc., are not recorded, since they don't result in a definite action. The DICT and PAD are lost at the termination of a Retrieval execution.

I shall now go on to explain the use of DICT and PAD by means of a sample execution; two commands I shall use that you haven't met before are 'DICT ALL' and 'PAD ALL', which cause all of the DICT and PAD respectively to be displayed. First however, I must explain a few of the labels used in these displays. (Note: The labels I don't mention eg. Q1, Q2, RS, don't concern us yet.)

DICT format

- The first 2 digit number is simply the number order on DICT. It is the 'STRING' number printed out following a RETAIN.

Type: T = The use to which the string has been put. When T = TERM the string is yet to be used, but it will be changed after a SEARCH to T = SEAR.

Origin: O = The source from which the string originated:-

O = SYST (for "System") when an actual Subject Index term is selected (eg. RETAIN 4567).

O = USER when string details are stored by means of the unqualified RETAIN (commonly following a MYUDC input).

PDP = The starting place on the PAD in which the results of any search using this string are written.

ASM = (when 0 = SYST) the reference number (eg. if RETAIN 0344 then ASM = 0344).

In connection with the PDP label, I must now tell you that two entries are made on the PAD for each SEARCH - one for 'direct hits' and the next for 'root hits'; so if PDP = 02 say (for T = SEAR) then PDP = 03 will also be relevant to this DICT item. If PDP = 00, it means (for T = TERM) that a SEARCH has not yet been performed. The relevant UDC string is also held in the DICT, and this appears on the second line of each separate DICT entry.

PAD format

- The first two digit number is the number order on the PAD (and is that number referenced by the PDP label in the DICT).

Search Type Flag: STF = 0 if the entry concerns direct hits,
= 1 if the entry concerns root hits (this is a simplification but it will do for now).

HITS = The same hit figure that is printed as a result of a SEARCH, type (direct or root) depending upon 'STF'.

I guess some of this must look a bit obscure at the moment, but I can assure you that its relevance will become obvious. Now for the example; try following it through line by line; I've simply copied the computer printout, but notes have been added (bracketed numbers) :

```
>@XQT FILMLIB.RETRIEVE

ENTER COMMAND
>TERM
ENTER TERM
>PARKER
   *REF:- 0199  PARKER COMMISSION ON NORTHERN IRELAND INTERNMENT
                341.345 (411) 06.045 PARKER
>RETAIN 0199
SYSTEM STRING 01 IS 341.345 (411) 06.045 PARKER

ENTER COMMAND
>DICT ALL
   01  T=TERM  O=SYST  ASM=0199  Q1=00  Q2=00  PDP=00
   341.345 (411) 06.045 PARKER

ENTER COMMAND
>PAD ALL ..................................... (1)

ENTER COMMAND
>SEARCH
0001 DIRECT HITS
0000 ROOT HITS

ENTER COMMAND
>DICT ALL
   01  T=SEAR  O=SYST  ASM=0199  Q1=00  Q2=00  PDP=01
   341.345 (411) 06.045 PARKER

ENTER COMMAND
>PAD ALL
   01  RS=0  STF=0  HITS=0001
   02  RS=0  STF=1  HITS=0000

ENTER COMMAND
>MYUDC
ENTER UDC STRING
>341

ENTER COMMAND
>RETAIN
USER STRING 02 IS 341

ENTER COMMAND
>DICT ALL
   01  T=SEAR  O=SYST  ASM=0199  Q1=00  Q2=00  PDP=01
   341.345 (411) 06.045 PARKER
   02  T=TERM  O=USER  ASM=0000  Q1=00  Q2=00  PDP=00
   341

ENTER COMMAND
>SEARCH
0001 DIRECT HITS
3035 ROOT HITS

ENTER COMMAND
>DICT ALL
   01  T=SEAR  O=SYST  ASM=0199  Q1=00  Q2=00  PDP=01
   341.345 (411) 06.045 PARKER
   02  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=03
   341
```

```
ENTER COMMAND
>PAD ALL
   01  RS=0  STF=0  HITS=0001
   02  RS=0  STF=1  HITS=0000
   03  RS=0  STF=0  HITS=0001
   04  RS=0  STF=1  HITS=3035

ENTER COMMAND
>CURRENT ..................................... (2)
CURRENT DICT NO:- 02

ENTER COMMAND
>DICT 2 ...................................................
   02  T=SEAR  O=USER  ASM=0000  01=00  02=00  PDP=03
   341

ENTER COMMAND
>RESET 1 ..................................... (4)
CURRENT DICT NO:- 01

ENTER COMMAND
>DICT ......................................................
   01  T=SEAR  O=SYST  ASM=0199  01=00  02=00  PDP=01
   341.345 (411) 06.045 PARKER

ENTER COMMAND
>PAD
   01  RS=0  STF=0  HITS=0001

ENTER COMMAND
```

Notes

(1) The PAD is empty (no searches have yet been made), so there is nothing to display.

(2) This command requests that the 'Current DICT Number' (the sequence number on DICT) be printed.

(3) This form of the 'DICT' command (DICT¥02 or DICT¥2) requests that details of a specific DICT item be printed. Ditto for the 'PAD' command.

(4) Referring back to Note (2) it will be seen that I used the phrase "Current DICT Number", by which I mean the most recently referenced entry on the DICT. As the Retrieval execution continues, the Current DICT Number will go on increasing (as new additions are made to the DICT) but if I want to go back and do some work using an earlier DICT entry, then I use the 'RESETØN' command, which will position me at the specified item number (N) on the DICT - eg. RESET 01 (or RESET 1) causes DICT entry number one to become the CURRENT DICT Number. If I then make a new entry (by means of RETAIN for example) the system will add one to my previous highest DICT number, and insert it there - the new entry then becoming the Current DICT Item (i.e. there is no danger of losing DICT entries by overwriting).

PAD is reset in accordance with DICT (again with protection against overwriting) so that the 'Current PAD Item' = PDP of the Current DICT Item after a RESET. A further example follows:-

```
ENTER COMMAND
>CURRENT
CURRENT DICT NO:- 11

ENTER COMMAND
>RESET 5
CURRENT DICT NO:- 05

ENTER COMMAND
>RETAIN 0083
SYSTEM STRING 12 IS 341.225.1 (491.1)

ENTER COMMAND
>CURRENT
CURRENT DICT NO:- 12

ENTER COMMAND
```

(5) The 'DICT' command with no argument (i.e. qualifying number) assumes that details of the DICT entry for the Current DICT Number are being elicited which, in this case (thanks to the preceding RESET), is number one. The same applies to the 'PAD' command.

Qualification Searching

Don't be put off by this heading, but rather consider what is to happen if a SEARCH results in an unwieldy number of UDC strings that satisfy the RETAIN string (eg. the 3036 hits in the last example but one, where UDC=341). If this large number is made up of 'direct hits', then you have no choice but to reformulate the enquiry to yield a more specific term; but if it contains a large number of 'root hits', then you can examine the additional information contained in the longer (more specific) root hit UDC strings. In the case of the 3035 root hits starting with the digits 341 (International Law), if you really did want everything on International Law then you would be left to ruminate on 3036 items (including the one direct hit), but if you were only interested in International Law with respect to people (.007), wouldn't it be convenient to select only those strings, of the 3035, that also contain the .007 classification?

It would hardly have mentioned this if the computer was unable to do it and so, needless to say, it is a facility of the Retrieval System. Because strings are subjected to further qualification (in the above example they must contain a .007 as well as starting with a 341) I have called the process 'Qualification Searching', and the relevant command is the shortened 'QSEARCH'. Now, 'Q-searching' is responsible for the third lot of note keeping that I mentioned earlier (the other two types being DICT and PAD), but before I go on to elucidate on this, I should just mention 3 further characteristics of Q-searching:

(i) The 'Qualifier' (.007 above) can appear anywhere in the string being Q-searched, unless you desire otherwise - see (iii) below.

The system protects against any ambiguity which might arise. If the Qualififer were 63 say, then in:

341(421.2)636.1 - qualification would be successful whereas in:

341(742)263.2 - qualification would be unsuccessful due to the ambiguous occurrence of the 63 'sub-string'.

(ii) The Qualifier is "SCORED' against the string being Q-searched, in the same way that scores are printed under the BROWSE command (see earlier),

e.g

STRING	QUALIFIER	SCORE
341.2(421)	(421)	100%
341.2(421.2)	(421)	100%
341.2(421)	(422)	75%
341.2(421.2)	(43)	66%

Also, the highest score in the string is that which is recorded,

eg: Q-searching-341:340.96

with Qualifier-340.9

would score the 341 part at 40%

but would score the 340.96 part at 100%

so the latter score would be the one that is kept.

The importance of scoring is that you can set a 'Threshold' to indicate that you are only interested in strings having a 'qualification score' equal to or greater than the Threshold value. For example, if you set a Threshold of 90% (typing THRESHØ90 would do this) then in the String-Qualifier-Score table above, only the first two strings would be kept for your reference.

(iii) During the process of Qsearching, the computer asks the user if the Qualifier is to be situated in the same facet as the UDC string

used in the previous search. This is known as "facet constraint" and I shall postpone discussion of its importance until the last of the Qsearching examples.

Keeping notes on Q-searches

In fact, details of Q-searches are kept on both the DICT & PAD in much the same way as conventional command and search details, but the Qsearched UDC strings (those above the Threshold value that is) and SCORE details are kept in a place called the 'RESULT' area, from which information can be read (eg. 'RESULT/ALL' causes the whole of the RESULT area to be displayed).

How to QSEARCH

In order to instigate a qualification search, one needs:-

(i) A qualifier, and

(ii) A set of strings to which qualification is to be applied.

To select a qualifier, you can use MYUDC, EDIT or TERM followed by RETAIN; that is, you can use any of the methods by which a UDC string is selected for a standard SEARCH. Try following the example below:

```
>@
END OF EXECUTION
>@XQT FILMLIB.RETRIEVE

ENTER COMMAND
>TERM
ENTER TERM
>NORMANS
    *REF:- 0298   NORMANS
                   (=15)
>RETAIN 0298
SYSTEM STRING 01 IS (=15)

ENTER COMMAND
>EDIT
-(=15)
>    3
(=13)

ENTER COMMAND
>RETAIN
USER STRING 02 IS (=13)
```

```
ENTER COMMAND
>MYUDC
ENTER UDC STRING
>.007

ENTER COMMAND
>RETAIN
USER STRING 03 IS .007

ENTER COMMAND
>DICT ALL
   01  T=TERM  O=SYST  ASM=0298  Q1=00  Q2=00  PDP=00
   (=15)
   02  T=TERM  O=USER  ASM=0000  Q1=00  Q2=00  PDP=00
   (=13)
   03  T=TERM  O=USER  ASM=0000  Q1=00  Q2=00  PDP=00
   .007
```

Having selected a qualifier, you must have access to a group of UDC strings to be qualified, which most probably will be a large set of roots hits, eg:

```
ENTER COMMAND
>MYUDC
ENTER UDC STRING
>341.1

ENTER COMMAND
>RETAIN
USER STRING 04 IS 341.1

ENTER COMMAND
>SEARCH
0001 DIRECT HITS
0567 ROOT HITS
```

You now have the two necessary ingredients for a Q-search to proceed, but before issuing the command you should set a Threshold value in order to reject strings that won't be of interest. Given the qualifier .007 for example:

Threshold Score %	Eligible strings must contain
100	.007
75	.00
50	.0

Example:

```
ENTER COMMAND
>THRESH 75

ENTER COMMAND
>QSEARCH BY 03 ON 04
QSEARCH WITH THRESHOLD OF 075 ? (YES OR NO)
>YES
FACET CONSTRAINT ON 04? (YES OR NO)
>NO

ENTER COMMAND
>DICT
   03  T=QSEA  O=USER* ASM=0000  Q1=04  Q2=04  PDP=03
   .007

ENTER COMMAND
>PAD
   03 - RS=1  STF=1  HITS=0002

ENTER COMMAND
>RESULT
   0001  REL=075%  ABS=075%  341.123:061:577.4.001.5
   0002  REL=075%  ABS=075%  341.123.001.5:633
```

Action: Set a Threshold of 75% Q-search by Qualifier 03, which is a sub-string.007 (03 referring to a DICT entry) on the set of root hits connected with DICT Item 04 (see previous example). The results of the QSEARCH are written as the Current Items on the DICT and the PAD, and are displayed by simply typing 'DICT' and 'PAD' (the default rule assumes the Current Item is required in each case). The results of the QSEARCH are displayed from where they have been written in the RESULT area by typing the RESULT command - the default rule again ensures that only the area relevant to the latest QSEARCH is displayed. All the items displayed in this way contain the .00 group (thanks to the 75% Threshold).

A Q-seach could cover more than one set of UDC strings, for example:

QSEARCH BY 09 ON 11 TO 18

would be a valid command if DICT items 11 thru 18 have 'Q-searchable sets' associated with them. A group of root hits is a Q-searchable set, and the results of a QSEARCH also constitute a Q-searchable set (eg. RESULT items 0001 & 0002 in the previous example). You could, if you wished, do the following:-

QSEARCH BY 03 ON 03

Refers to the qualifier .007 again

Refers to the set of results that was created by the previous Q-search using qualifier .007

Now the results of a QSEARCH are stored with the qualifier details on the DICT, but the above command would necessitate writing a new DICT item to store details of the new QSEARCH (DICT item 03 is already full of details from the QSEARCH BY 03 ON 04). This point is illustrated by the following example:-

```
ENTER COMMAND
>THRESH 0

ENTER COMMAND
>DICT ALL
   01  T=TERM  O=SYST  ASM=0298  Q1=00  Q2=00  PDP=00
   (=15)
   02  T=TERM  O=USER  ASM=0000  Q1=00  Q2=00  PDP=00
   (=13)
   03  T=QSEA  O=USER  ASM=0000  Q1=04  Q2=04  PDP=03
   .007
   04  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=01
   341.1

ENTER COMMAND
>QSEARCH BY 03 ON 03
QSEARCH WITH THRESHOLD OF 000 ? (YES OR NO)
>YES
NEW DICT ENTRY NO:- 05
FACET CONSTRAINT ON 03? (YES OR NO)
>NO

ENTER COMMAND
>PAD
   04  RS=1  STF=1  HITS=0002

ENTER COMMAND
>DICT ALL
   01  T=TERM  O=SYST  ASM=0298  Q1=00  Q2=00  PDP=00
   (=15)
   02  T=TERM  O=USER  ASM=0000  Q1=00  Q2=00  PDP=00
   (=13)
   03  T=QSEA  O=USER  ASM=0000  Q1=04  Q2=04  PDP=03
   .007
   04  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=01
   341.1
   05  T=QSEA  O=USER  ASM=0000  Q1=03  Q2=03  PDP=04
   .007

ENTER COMMAND
>RESULT
   0003  REL=056%  ABS=075%  341.123:061:577.4.001.5
   0004  REL=056%  ABS=075%  341.123.001.5:633

ENTER COMMAND
>@
END OF EXECUTION
```

The 'ABS' field ("Absolute Score") applies to each individual QSEARCH, whereas the 'REL' field ("Relative Score") is a cumulative measure - 75% of 75% = 56%. When Q-searching, the threshold always applies to the Relative Score, so that Q-searching on the results of a previous QSEARCH can be used to refine the set of results. For example, given 10,000 root hits in a set S1, suppose:

a QSEARCH by .007 on set S1 gives 1,000 items at 100% (S2)

a QSEARCH by "1948" on set S2 gives 10 items at 100% (S3)

a QSEARCH by (43) on set S3 gives 10 items at 100%

These last 10 items will contain .007, "1948" and (43)

Repeated Q-searching causes the Absolute and Relative Scores to diverge (after only one QSEARCH, Absolute and Relative Scores are always equal) eg:-

Set Resulting from this QSEARCH	QSEARCH BY	ON	ABS	REL
(1)	.007	34.007(43)"1926"	100	100
(2)	(43)	Set (1)	100	100
(3)	(44)	Set (1)	50	100%×50% = 50%
(4)	"1948"	Set (3)	50	50%×50% = 25%

It can be seen from the DICT display in the previous example that 'Q1' and 'Q2' refer to the range of DICT items that were Q-searched, eg:

QSEARCH BY 09 ON 06 TO 08

would give (for 'QSEA' DICT Item 09) Q1 = 06 Q2 = 08;

and QSEARCH BY 12 ON 10

would give (for 'QSEA' DICT Item 12) Q1 = 10 Q2 = 10

DICT items Q1 thru Q2 should, of course, be valid 'Q-searchable' sets.

Facet Constraint

The Qsearching examples I have given so far have all involved a "NO" response to the "facet constraint" question. A "YES" response on the other hand, restricts the Qsearch to only those facets that contain the string that satisfied the previous search. Try an example (the facet constraint will isolate only those UDC numbers having 341 & .007 in a single facet):-

```
>@XQT FILMLIB.RETRIEVE

ENTER COMMAND
>THRESH 100

ENTER COMMAND
>MYUDC
ENTER UDC STRING
>341

ENTER COMMAND
>RETAIN
USER STRING 01 IS 341

ENTER COMMAND
>SEARCH
0001 DIRECT HITS
3035 ROOT HITS

ENTER COMMAND
>MYUDC
ENTER UDC STRING
>.007

ENTER COMMAND
>RETAIN
USER STRING 02 IS .007

ENTER COMMAND
>QSEARCH BY 02 ON 01
QSEARCH WITH THRESHOLD OF 100 ? (YES OR NO)
>YES
FACET CONSTRAINT ON 01? (YES'OR NO)
>NO

ENTER COMMAND
>PAD
   03   RS=1   STF=1   HITS=0009
```

```
ENTER COMMAND
>RESULT
   0001  REL=100%  ABS=100%  341.231 U.D.I.(674)321.21.007(=96)
   0002  REL=100%  ABS=100%  341.231 UDI (674)343.261 (674)07.007
   0003  REL=100%  ABS=100%  341.231 UDI (674)343.264:07.007
   0004  REL=100%  ABS=100%  341.231 UDI (674)663.97.05:631.007.1
   0005  REL=100%  ABS=100%  341.232.2: 322.1.007.1
   0006  REL=100%  ABS=100%  341.345.007 (41)
   0007  REL=100%  ABS=100%  341.358.007
   0008  REL=100%  ABS=100%  341.358.007
   0009  REL=100%  ABS=100%  341.358.007

ENTER COMMAND
>QSEARCH BY 02 ON 01
QSEARCH WITH THRESHOLD OF 100 ? (YES OR NO)
>YES
NEW DICT ENTRY NO:- 03
FACET CONSTRAINT ON 01? (YES OR NO)
>YES

ENTER COMMAND
>PAD
   04  RS=1  STF=1  HITS=0004

ENTER COMMAND
>RESULT
   0010  REL=100%  ABS=100%  341.345.007 (41)
   0011  REL=100%  ABS=100%  341.358.007
   0012  REL=100%  ABS=100%  341.358.007
   0013  REL=100%  ABS=100%  341.358.007

ENTER COMMAND
>@
END OF EXECUTION
```

Having written an explanation thus far, let me try to reassure you that it is, in practice, much easier than it looks. A first sight of 'Qualifiers', 'Q-searchable sets', 'Thresholds' etc., probably makes it all look a bit daunting, but this jargon will fall into place when you try it yourself, and things might seem easier if you take a look at the sample executions contained in Appendix D.

Save Sets

There is another type of Q-searchable set(the other two types being Root Hits and QSEARCH Results); it is known as a 'Save Set' and is denoted on the DICT by T=SAVE, O= UDC. A 'Save Set' is simply a group of UDC strings taken from the UDC catalogue, that can then be Q-searched. 'SAVE' is a low level command that can only be issued when browsing in the UDC file (i.e. it must follow the BROWSE command). In 'SAVE␣N', N is the number of UDC strings being saved, inclusive of the string currently displayed. N must be between 1 and 999. 'SAVE' on its own (with no argument) is interpreted as 'SAVE 1'.

The use of the SAVE command is best illustrated by example:-

```
>@XQT FILMLIB.RETRIEVE

ENTER COMMAND
>MYUDC
ENTER UDC STRING
>[BBC]

ENTER COMMAND
>RETAIN
USER STRING 01 IS [BBC]

ENTER COMMAND
>SEARCH
0000 DIRECT HITS
0016 ROOT HITS

ENTER COMMAND
>BROWSE
COMMENCE AT FIRST ROOT HIT:- 0235/12
  SCORE=100  UDC= [BBC] 654.19
>SAVE
SAVE SET STORED ON DICT NO:- 02
>J 10
  SCORE=100  UDC= [BBC] 654.19:62.007
>SAVE 20
SAVE SET STORED ON DICT NO:- 03
>J 50
  SCORE=050  UDC= [BRITISH NUCLEAR FUELS LTD]
>SAVE 999
SAVE SET STORED ON DICT NO:- 04
>DICT ALL
   01  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=01
   [BBC]
   02  T=SAVE  O=UDC   ASM=0000  Q1=00  Q2=00  PDP=03
   [BBC]
   03  T=SAVE  O=UDC   ASM=0000  Q1=00  Q2=00  PDP=04
   [BBC]
   04  T=SAVE  O=UDC   ASM=0000  Q1=00  Q2=00  PDP=05
   [BBC]

ENTER COMMAND
>PAD ALL
   01  RS=0  STF=0  HITS=0000
   02  RS=0  STF=1  HITS=0016
   03  RS=0  STF=1  HITS=0001
   04  RS=0  STF=1  HITS=0020
   05  RS=0  STF=1  HITS=0999

ENTER COMMAND
```

The UDC string associated with the Save Set on the DICT-[BBC] above - is pretty arbitrary, it is simply the nearest UDC string to hand when BROWSing commences and so it may, or may not, be meaningful.

A QSEARCH BY 05 ON 01 TO 04 (assuming DICT 5 to be a qualifier) would initiate a QSEARCH on all the 'saved' UDC numbers, storing the results:

(i) in DICT item 05 (Q1 = 01, Q2 = 04, PDP = 06)

(ii) in PAD item 06

(iii) and in the RESULT area

Note (i) All 3 types of Q-searchable set are denoted on the PAD by setting 'STF' (the "Search Type Flag") to 1. A summary of the results of a QSEARCH is recorded on the PAD, and the item 'RS' is set to 1 to denote that this PAD entry refers to the RESULT area (where scores & strings themselves are stored).

(ii) You will have noticed that Direct Hits don't qualify to be a Q-searchable set. This is simply because the characteristics of a Direct Hit are precisely defined (they match a known search term) and their information content beyond this is zero.

Facet Searching

There is one last form of searching at your disposal, which is both easy to handle and potentially of great use.

All the UDC strings that can be acessed by a startndard SEARCH are also split up into their separate facets, and so it is possible to select a UDC string - by TERM, EDIT or MYUDC followed by RETAIN - and then search for any occurrence of that string anywhere in the main UDC file, thereby circumventing the need to re-order strings and re-file them in several places in the catalogue. This process is known as "facet searching" and is instigated by the command FSEARCH. The file that is then searched contains only individual facets, so the current term (that used in the FSEARCH) should itself be free of complexity - ie. it should consist of one facet only.

Results are printed in the same way as a standard SEARCH, and a BROWSE can follow an FSEARCH, but the strings browsed will not necessarily be in UDC order, since when the strings are split into facets to form the "facet search file", the conventional UDC ordering is lost.

On the DICT, an item that has been used for an FSEARCH is characterized by T = FSEA, and a 'save set' constructed following an FSEARCH has O = FCT. When an FSEARCHed item is QSEARCHed, all hits are considered (ie. both Direct and Root Hits of an FSEARCH constitute a Q-searchable set).

Example:

```
>@XQT FILMLIB.RETRIEVE

ENTER COMMAND
>MYUDC
ENTER UDC STRING
>343.77

ENTER COMMAND
>RETAIN
USER STRING 01 IS 343.77

ENTER COMMAND
>DICT ALL
   01  T=TERM  O=USER  ASM=0000  Q1=00  Q2=00  PDP=00
   343.77

ENTER COMMAND
>FSEARCH
0090 DIRECT HITS
0000 ROOT HITS

ENTER COMMAND
>DICT ALL
   01  T=FSEA  O=USER  ASM=0000  Q1=00  Q2=00  PDP=01
   343.77

ENTER COMMAND
>PAD ALL
   01  RS=0  STF=1  HITS=0090
   02  RS=0  STF=1  HITS=0000

ENTER COMMAND
>BROWSE
COMMENCE AT FIRST ROOT HIT:- 0726/06
  SCORE=083  UDC= (489 COPENHAGEN) 73.027: 343.78
>N-3
  SCORE=100  UDC= 34.096:343.77(421.4 GUILDFORD)(421.281.5)
  SCORE=100  UDC= 34.096:342.722(411)343.77(421.2)"8.3.73"
  SCORE=100  UDC= 34.096:343.77
>SAVE
SAVE SET STORED ON DICT NO:- 02
```

```
>DICT ALL

   01  T=FSEA  O=USER  ASM=0000  Q1=00  Q2=00  PDP=01
   343.77
   02  T=SAVE  O=FCT   ASM=0000  Q1=00  Q2=00  PDP=03
   343.77

ENTER COMMAND
>RESET 1
CURRENT DICT NO:- 01

ENTER COMMAND
>SEARCH
NEW DICT ENTRY NO:- 03
0000 DIRECT HITS
0000 ROOT HITS

ENTER COMMAND
>FSEARCH
NEW DICT ENTRY NO:- 04
0090 DIRECT HITS
0000 ROOT HITS

ENTER COMMAND
>DICT ALL
   01  T=FSEA  O=USER  ASM=0000  Q1=00  Q2=00  PDP=01
   343.77
   02  T=SAVE  O=FCT   ASM=0000  Q1=00  Q2=00  PDP=03
   343.77
   03  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=04
   343.77
   04  T=FSEA  O=USER  ASM=0000  Q1=00  Q2=00  PDP=06
   343.77

ENTER COMMAND
>PAD ALL
   01  RS=0  STF=1  HITS=0090
   02  RS=0  STF=1  HITS=0000
   03  RS=0  STF=1  HITS=0001
   04  RS=0  STF=0  HITS=0000
   05  RS=0  STF=1  HITS=0000
   06  RS=0  STF=1  HITS=0090
   07  RS=0  STF=1  HITS=0000

ENTER COMMAND
>@
END OF EXECUTION
>@FIN


  RUNID: ROSTFF    ACCT: SCS              PROJECT: BSTAFF
  TIME:     TOTAL: 00:03:57.240      CBSUPS:  037097185
            CPU:   00:01:13.399      I/O:   00:02:03.887
            CC/ER: 00:00:39.953      WAIT:  00:24:55.097

  SUAS USED:  $    25.47   SUAS REMAINING:  $  9974.28
  IMAGES  READ:   211        PAGES:  31
  START:    10:29:02 DEC 07,1977    FIN:  10:58:35 DEC 07,1977
*TERMINAL INACTIVE*
>@@TERM
```

APPENDIX A - Connecting the Terminal to Milton Keynes

(1) Plug in and switch on both the terminal and the Modem (the grey box on the floor) at the wall. Press the "ON" switch to the left of the terminal keyboard.

(2) Of the 8-buttons to the left of the keyboard, only the "FDX/.HDX" and the "300"-buttons should be depressed - the others should be standing up.

(3) On the keyboard itself, the "CAPS LOCK" key should be depressed.

(4) Pick up the phone headset and dial 0908 63984. Tell whoever answers that you're phoning from the BBC Film Library and you want to be connected to the computer. When you hear a continuous whistle (sometimes you'll get this as soon as you've dialled the number), press the "DATA" on the top of the phone (the leftmost button) and put the headset down next to the phone - don't replace the headset!

(5) Press the "RETURN" key. The computer will respond with:

SCICON/OPEN UNIV. FRONT END SYSTEM.....etc.....
UNIVAC SITE ID :

(6) Type UHI207 (O=zero). Don't press RETURN, simply wait for the computer to respond with:

*UNIVAC 1100 OPERATING SYSTEM.....etc.....

If no such response is forthcoming, try typing UHI207 again. If still nothing happens, hang up the phone and go back to step (4).

(7) If all has gone well you are now connected to the computer, and you should continue by typing the @RUN statement - see Section 2, "Getting at the Retrieval System".

APPENDIX B - Errors and Unexpected Messages

At the time of writing this piece, I'm not sure of all the errors and unexpected contingencies that can arise. Error messages within the Retrieval System itself are generally self-explanatory, but those that may occur at other stages (eg. when signing on, when assigning files etc.) can be next to useless.

The errors that are most difficult to talk about are those detected by the EXEC, since they are many and varied, and they can crop up at any time (even from within the Retrieval System - the execution of which will be immediately terminated in such circumstances). All I can do therefore, is to list a few of the errors and unexpected contingencies that you may encounter, together with some intimation of when you might expect them and what effect they have.

If you have any problems, don't hesitate to ask me if I'm around, or phone me on:

0908 63188 and ask for Brian Staff

Please keep printout on which errors occur, so that I can investigate their cause.

Important !

The phone line to Milton Keynes is not perfect (far from it), and sometimes the telephone will add errors willy-nilly. As a general rule therefore, when you get an error or an unexpected message, you should simply try again, just to make sure that it's not the phone to blame. Genuine errors will occur however, and these are outlined below according to the stage at which they can occur:-

When signing on

Having established the phone link (see Appendix A), pressing carriage return may result in the ever depressing message:

FE MSG UNIVAC DOWN

Whenever you receive this message, the only thing to do is hang up the phone and try again later.

When entering the @RUN statement

(1) As long as the computer replies with "DATE: ... and TIME: ..." information, you've nothing to worry about. Additional messages (eg. 'DUP ID') are no problem as long as the Date and Time are printed.

(2) The message

NO RUN ACTIVE

simply means that an @RUN statement is expected, and if you wish to start a run you should enter a correct @RUN statement. If you don't want to carry on, you should type '@@TERM' and hang up the phone.

(3) The message:

SYSTEM HOLD ON DEMAND RUNS

means that no terminals (known as "demand users") are allowed to access the computer. Such a condition will generally be maintaine until the computer is less busy, so you should type @@TERM and hang up.

(4) The message:

ILLEGAL ACCOUNT NUMBER

can mean one of two things:-

(i) you have mistyped the @RUN command, and the word 'SCS/BS' has not been intepretted as such, or

(ii) only users having selected (priority) account numbers are being allowed to use the computer.

If you can assure yourself that you are not guilty of (i), you should type @@TERM and hang up. Try again later.

After typing the @ADD command

(1) A reply of : ADD ELEMENT NOT FOUND will most probably mean that you have typed the command incorrectly.

(2) If your @ADD command is printed back at you, together with a message such as : ERROR IN ADD ELEMENT, it could mean that you've mis-typed the command. If this is not the case however, it usually means that the @RUN statement was incorrectly interpreted (the telephone may have caused this), so you should type @FIN and press RETURN. This will give you the accounting information, and the rightmost phrase on the top line should be "PROJECT: BSTAFF". If a name other than BSTAFF appears, it signifies that the RUN statement was mistaken, so you should type it again (@RUN..etc..) and then try the @ADD FILMLIB.GO command afresh.

(3) To a message of : WAITING ON FACILITY, you should type : @@X/T (all other replies will result in a *WAIT LAST INPUT IGNORED* message). The '@@X T' command will result in the message:

EXECUTION TERMINATED

This sequence of events is caused by the unavailability of an information file - contact me and I'll sort it out.

(4) Any other unusual response may or may not preclude operation of the retrieval system. You can always try it out by sending:

@XQT FILMLIB.RETRIEVE

which should give a reply of ENTER COMMAND. If it doesn't, or if you're in any doubt about the response you get after typing the @ADD command, you should contact me.

After any command beginning with the '@' symbol (i.e. EXEC commands)

Possible messages are:

PROGRAM NOT FOUND or

SYNTAX ERROR

Such a response invariably means that you have mistyped the command - just try again.

During Retrieval System execution

Certain messages (some errors, some notifications of illegal operations) will be printed out by the Retrieval System during execution, which (though they might have curious effects) will not terminate execution. The following messages however, lead, without fail, to EXEC intervention and program termination. Please contact me if such errors occur, and keep the relevant printout:

(i) IGDM + other information

(ii) IOPR + other information

(iii) I/O ERROR + other information

(iv) ERROR ENCOUNTERED WHEN PROCESSING LABELS .....

(v) ATTEMPT TO READ BEYOND END OF FILE .....

If you get an error of the type:

NO EQUIPMENT ASSIGNED TO FILE ...

it may well means that you haven't assigned the files (by the @ADD command) and having remedied this, you will be able to execute the program properly.

Unexpected messages at any time

The computer operator is able to send messages to users at any time. Such messages will appear in the form:

*TB ... message .....

Generally, such messages will contain information being sent to all users and will not necessarily demand any action on your part, eg:

*TB DISC PROBLEMS CAUSING SLOW RESPONSE

is simply an explanation of poor performance. The message:

*TB REBOOTING IN 5 MINUTES

means that the computer will be stopped in five minutes, so you might as well stop program execution and sign off. Then there is the more explicit:

*TB PLEASE FIN

or some other message to denote that you should leave the computer. An '@FIN' can be entered whenever the program is expecting input (i.e. whenever the '>' character has been printed) eg:- immediately after the @RUN statement; before program execution; during program execution etc. (When the program is not expecting input the 'BREAK' key and/or '@@XØTIO' commands may be necessary beforehand, see SECTION 2 - "Stopping the Retrieval System").

As soon as accounting and timing information has been printed, it means that you have left the computer and the operator will be satisfied. To break the phone line (to save on phone bills) you should then type @@TERM and hang up.

If the computer operator asks you a direct question, eg:

*TB PLEASE IDENTIFY YOURSELF

you can reply <u>at any time</u> (providing that the terminal is not printing) by typing:

@@MSGØ ... message ...

e.g. @@MSGØ I AM ALBERT EINSTEIN AT THE BBC FILM LIBRARY.

If the operator demands more technical information than you are able to supply, you can either type:

@@MSG I DON'T UNDERSTAND, PLEASE PHONE ...

or you can contact me and I'll sort it out.

(NOTE: the @@MSG command does not terminate program execution. After sending a message to the operator in this way you can carry on as before).

If all computer response ceases

Try the following steps in turn:

(i) Hold down the CONTROL key and press the X key. If you get a reply, carry on as normal.

(ii) Press the RETURN key repeatedly. If this results in a WAIT message, you can either wait, or you can terminate the run - see Section 2, "Stopping the Retrieval System".

(iii) Type a single @ and press RETURN.

(iv) Type @@X/TIO (/=space, O= the letter) and press RETURN.

If all this fails to provoke any response, it means either that you've lost the phone line (hang up and try again) or that the computer has broken down (hang up and re-dial 0908 63984 - whoever answers will tell you if the computer is working).

APPENDIX C - Retrieval Commands

High Level Commands can be entered at (almost*) any time, assuming, of course, that the computer is ready, whereas Low Level Commands can only be entered from within direct, file-referencing high level commands (TERM, EXPAND, AFILE and BROWSE). Low level commands can never be used in reply to the 'ENTER COMMAND' message.

*When explicit information is required, it must be given (neither high nor low level commands will be understood at such times). Such explicit information is required by:-

(i) TERM : whatever you type in following the 'ENTER TERM' message will be taken as a search literal for the Subject Index.

(ii) MYUDC : whatever you type in reply to the 'ENTER UDC STRING' message, the Retrieval System will assume to be a valid UDC number.

(iii) AFILE SEARCH : as for MYUDC.

(iv) EDIT : whatever you type on the correcting line is assumed to constitute editing instructions to be applied to the line above.

(v) QSEARCH : (a) following the 'QSEARCH WITH THRESHOLD OF X? (YES OR NO)' message, anything other than a 'YES' (or simply 'Y')input will immediately terminate the QSEARCH.

(b) following the "FACET CONSTRAINT...." message, a 'YES' (or simply 'Y') input will cause such a constraint to be imposed, whereas any other

input will be taken as a 'NO' (or 'N') response.

NB : At any stage when input is expected, a '@' character (in the first position on the line after the prompt '>') will cause execution to terminate (except when in the form @@MSG - See Appendix B).

High Level Commands

Command : TERM

Format : > TERM

Additional Information : A literal subject term solicited immediately following the TERM command.

Action : Search the Subject Index for the nearest match to the given subject literal. Display the nearest match and switch into Subject Browse Mode.

Command : EXPAND

Format (1) : > EXPAND

Valid : Only when in Subject Browse Mode

Action : Alphabetically expand (± 2) on the item in the Subject Index most recently displayed and remain in Subject Browse Mode.

Format (2) : > EXPAND ∅ N, where N is a one or two digit number referring to a DICT entry.

Valid : Only for DICT Item N, when O = SYST.

Action : Display the Subject Index item referred to by DICT Item N, and switch into Subject Browse Mode.

Note: DICT Item N is not Reset to be the Current DICT Item by this command.

Command : MYUDC

Format : > MYUDC

Additional Information : A string of valid UDC characters (A-Z, 0-9, : [/etc.) solicited immediately following the MYUDC command

Action : Accept a UDC number directly from the user.

Note : The system checks for valid characters (and complains about invalid ones) but not for valid sequences. It is up to the user to validate the string.

Command : EDIT

Format : > EDIT

Additional Information : The necessary editing details - see below.

Action: Print the UDC number that was recently connected with a MYUDC entry, a RETAIN ␣ NNNN, a previous EDIT, or a RESET (ie. relevant to the Reset DICT Item). Then, on the next line, solicit editing information character by character such that:

(i) ␣ (space) - leave the above character as it is,

(ii) ## - replace the above character by a space,

(iii) X - replace the above character by character X, where 'X' is any valid UDC character.

Display the edited string and keep it handy for further reference ie. for RETAIN or EDIT. Such a string (like a string entered by MYUDC) is kept intact until overwritten by a string connected with a subsequent command (ie. MYUDC, RESET or RETAIN␣ NNNN) and it is assumed to be the subject of an unqualified RETAIN if such a command is issued.

Command : RETAIN

Format (1) : > RETAIN ␣ NNNN, whereNNNN is a four digit number (exactly 4 digits).

Valid : Only when NNNN is a 'REF' number, such as that displayed when browsing in the Subject Index on the Authority File.

Action : Store details of term number NNNN on the DICT with T = TERM and 0 = SYST. Inform the user as to the sequence number of the new item on the DICT, and make this the Current DICT Item.

Format (2) : > RETAIN

Action : Take the last UDC number to which there was explicit reference (by MYUDC, RETAIN ß NNNN, EDIT or RESET) and store details of it on DICT with T = TERM and 0 = USER. Inform the user as to the sequence number of the new item or the DICT, and make this the Current DICT Item.

Command : CURRENT

Format : > CURRENT

Action : Display the sequence number (or "string number") of the Current DICT Item.

Command : RESET

Format : > RESET ß N, where N is a one or two digit number.

Valid : Only when N refers to an existing DICT entry.

Action : Make DICT Item number N the Current DICT Item.

Note : The PAD is always reset in line with DICT.

Command : SEARCH

Format : > SEARCH

Action : Search the main UDC file for Direct and Root Hit matches to the UDC string found as the Current DICT Item. Display the results of the search and write them on the PAD in the first available (ie. empty) locations. Notify the DICT that PAD entries exist (ie. complete the 'PDP' entry on the DICT). Of the two PAD entries producted by each SEARCH, the first concerns Direct Hits and the second Root Hits. RS=0 for both types (ie. results are not stored in the RESULTS area) and STF=1 for Root Hits only (denoting a

Q-searchable set). Set type T = SEAR on the DICT for the Current DICT item.

Command : FSEARCH

Format : > FSEARCH

Action : Search the 'facet search file' for Direct and Root Hit matches to the UDC string found as the Current DICT Item which, if the results are to be meaningful, should consist of only one facet. FSEARCH details are recorded as for SEARCH, but now T = FSEA on the DICT, and STF=1 on the PAD for both Direct and Root Hits.

Note : For both SEARCH and FSEARCH, if type T is not equal to TERM before the command is entered, then a new entry is made on the DICT and the user is notified.

Command : BROWSE

Format : > BROWSE

Valid : Only when the Current DICT Item is of the type T = SEAR or T = FSEA.

Action : Display the first Root Hit and switch into UDC browse mode. If there are no Root Hits, display last Direct Hit + 1. If there are no Direct Hits, display the nearest match to the search string. When a BROWSE follows an FSEARCH, the strings will not necessarily be in UDC order (they will be ordered by individual facets).

Command : AFILE

Format (1) : > AFILE ß NNNN ß ACROSS, where NNNN is a four digit number (exactly four digits).

Valid : Only when NNNN is a 'REF' number.

Action : Display the item referred to by number NNNN and switch into Authority File Browse Mode.

Format (2) : > AFILE ß NNNN ß UP

Action : Display item NNNN and find its first superior in the UDC hierarchy.

Additional Info : After a superior is found, pressing carriage return will cause the next superior to be located, and this can be continued until no superiors exist - a condition to which the user will be alerted. Rather than carriage return, any High Level command can be entered, or even a Low Level browse command (which brings about a switch to Authority File Browse Mode).

Format (3) : > AFILE ß NNNN ß DOWN

Action : Display item NNNN and find possible inferiors in the UDC hierarchy (by addition of characters 0123456789.-). UDC inferiors that exist on the Authority File are displayed together with the literal subject description. Literals appearing within UDC numbers cause termination.

Additional Info : After all routes have been explored, the user is required to supply a one or two digit number describing the single route upon which further extension is required. This can continue until no routes are displayed (which means none exist). The AFILE ... DOWN process can be terminated at any time, but only High Level commands will be obeyed.

Format (4) : > AFILE ß NNNN ß DA

Action : Identical to AFILE ... DOWN, but now the full set of valid UDC characters - including : ("[etc. - can be used to compose extensions.

Note : This form of 'AFILE.. Down' is not strictly UDC hierarchical.

Format (5) : > AFILE ⅋ SEARCH

Additional Info : A UDC number is solicited immediately following the command.

Action : Search the Authority File for the nearest match to the given UDC string. Display the nearest match and switch into Authority File Browse Mode. If the Return key is pressed in reply to UDC solicitation, then the UDC string that was the subject of the most recent of the following commands is used :- MYUDC, EDIT, RESET or RETAIN ⅋ NNNN.

Note : No check is made for valid UDC sequences, although individual characters are vetted.

Command : THRESH

Format (1) : > THRESH ⅋ N, where N is a number between zero and 100 (inclusive)

Action : Set a 'Threshold' so that :-

(i) as a result of a QSEARCH, no UDC file string with a Relative Score of less than the Threshold value will be written in the RESULT area, or

(ii) no result held in the RESULT area will be displayed under the 'RESULT' command when the Absolute Score is less than the Threshold value.

Format (2) : > THRESH

Action : Display the current Threshold.

Command : QSEARCH

Format (1) : > QSEARCH ␣ BY ␣ XX ␣ ON ␣ YY, where XX and YY are 2 digit numbers (exactly 2 digits) referring to existing DICT entries.

Valid : Only when YY refers to a Q-searchable set on DICT.

A Q-searchable set is a set of UDC strings of one of the following four types:-

(i) Root Hits. DICT item must have T = SEAR
If PDP = N, then item N + 1 on the PAD must have 'HITS > zero'.

(ii) The results of a previous QSEARCH. DICT item must have T = QSEA; Q1, Q2 and PDP > zero. For PAD(N), 'HITS' must be greater than zero.

(iii) A Save Set. DICT item must be of type T = SAVE.

(iv) Any results (Direct or Root) associated with an FSEARCH (T=FSEA).

Action : Make DICT item XX the Current DICT Item and perform a Qualification Search on the set of UDC strings referenced by DICT item YY, using the Qualifier that is DICT item XX. Write strings having supra - Threshold Relative Scores in the RESULTS area, together with the Scores themselves. Write the number of supra-threshold scores (ie. the number of strings written in the RESULTS area) in the 'HITS' entry on the PAD. Set RS=1 (denoting reference to the RESULT area) and set STF=1 (denoting a Q-searchable set). Complete DICT item XX (this item points to the relevant PAD entry) by setting Q1 and Q2 to YY, and change type to T = QSEA.

Note : If T is not equal to TERM prior to the present QSEARCH, a new DICT item is created (the user is informed of this). This will always happen when a particular Qualifier is used in more than one QSEARCH.

Format (2) : > QSEARCH ß BY ß XX ß ON ß YY ß TO ß ZZ

Action : Similar to Format (1), but now the QSEARCH (by qualifier XX) is performed on all DICT items YY thru ZZ (all of which must be eligible - see (i) to (iv) above, defining Q-searchable sets). The relevant DICT entry is completed with Q1=YY and Q2=ZZ.

Format (3) : QSEARCH ß BY ß XX ß ON ß ALL

Action : QSEARCH by qualifier XX on the whole UDC file (which is specially "saved" as a new DICT item for the purpose).

Additional Info : Always after the QSEARCH command :

- (i) the user is given the chance to opt out. This is to ensure against Q-searching with a low threshold, which can be very time consuming. A 'NO' reply at this point will prematurely terminate the QSEARCH.
- (ii) The user is also given the option to impose a facet constraint which, if desired, means that the present QSEARCH is limited to those facets that satisfied the previous search (be it a SEARCH, an FSEARCH or a QSEARCH). In this way, UDC entries can be isolated that contain single facets which satisfy more than one search string.

Command : DICT

Format (1) : > DICT

Action : Display details of the Current DICT Item (if DICT is not empty).

Format (2) : -> DICT ß N, where N is a one or two digit number
referring to an existing DICT entry.

Action : Display details of DICT item number N.

Format (3) : > DICT ß ALL

Action : Display the whole DICTionary (if not empty).

Command : PAD

Format (1) : > PAD

Action : Display details of the Current PAD Item (if PAD is not empty).
This is the PAD item equivalent to the Current DICT Item.
In the case of a DICT item of type T =SEAR or T=FSEA (which has b
SEARCHd or FSEARCHd to produce PAD entries) the Current PAD Item
is always that concerning Direct Hits.

Note : A 'RESET' command resets the PAD as well as DICT.

Format (2) : > PAD ß N, where N is a one or two digit number referring
to an existing entry on the PAD.

Action : Display details of PAD item number N.

Format (3) : > PAD ß ALL

Action : Display the whole PAD (if not empty).

Command : RESULT

Format (1) : > RESULT

Valid : Only when the Current DICT Item is of type T = QSEA.

Action : Display the RESULT area relevant to the Current DICT
Item (if this area is not empty).

Format (2) : > RESULT ß N, where N is a one or two digit number
referring to an existing DICT entry.

Valid : Only when DICT entry N is of type T = QSEA.

Action : Display the RESULT area relevant to DICT item N (if not empty).

Format (3) : > RESULT ß ALL

Action : Display the whole RESULT area (if not empty).

Command : TIME

Format : > TIME

Action : Print the present date and time.

Low Level Commands

Applicable in all browse modes

High Level Command	Browse made that follows this command
TERM	Subject Index
EXPAND	Subject Index
AFILE - ACROSS	Authority File
- UP	Authority File
- SEARCH	Authority File
BROWSE	UDC File

Command : N

Format (1) : > N

Action : Display next item

Format (2) : > N +

> N -

Action : Display next (+) or previous (-) item.

Format (3) : > N + X (or N ∅ X)

> N - X, where X is a one or two digit number between zero and 99.

Action : Display next (+) or previous (-) X items.

Command : J

Format : > J + X (or J ∅ X)

> J - X, where X is a one or two digit number between zero and 99.

Action : Jump X - 1 items forward (+) or backward (-) and display the Xth.

Command : BASE

Format : > BASE

Action : Return to the point in the file at which browse mode was entered.

Note : When browse mode is entered by AFILE ... UP, a BASE command causes a return to the item on the hierarchical level immediately below that at which browsing was commenced.

Applicable only when in UDC Browse Mode

(ie. following the BROWSE command).

Command : SAVE

Format (1) : > SAVE

Action : Save a reference to the item most recently printed, by creating new entries on DICT (T = SAVE, PDP = Relevant PAD Pointer) and on the PAD (HITS = 1, RS = 0, STF = 1 - denoting a Q-searchable set), thus creating a set that can be Q-searched.

Format (2) : > SAVE ß N, where N is a number between 1 and 999.

Action : Save a reference to the item most recently printed and the next N - 1 items as well, by creating new entries on DICT and PAD (as in Format (1) but now HITS = N).

APPENDIX D - Complete Examples

```
SCICON/OPEN UNIV. FRONT-END SYSTEM  8-DEC-77  15:26   LINE - 10

 DEMAND TERMINALS NOW AVAILABLE

UNIVAC  SITE-I/D: UHI207
*UNIVAC 1100 OPERATING SYSTEM  VER. 33R2-0U16   (RSI)*
>@RUN ROSTFF,SCS/BS,BSTAFF
DATE: 120877       TIME: 152633
>@ADD FILMLIB.GO
READY
READY
READY
READY
READY
READY
READY
FURPUR27R2-4 72-8.1 12/08/77 15:49:35
   175 BLOCKS COPIED.
   265 BLOCKS COPIED.
    17 BLOCKS COPIED.
    17 BLOCKS COPIED.
    17 BLOCKS COPIED.
   109 BLOCKS COPIED.
READY
READY
READY
READY
READY
READY
>@XQT FILMLIB.RETRIEVE

ENTER COMMAND
>TERM
ENTER TERM
>FORENSIC
   *REF:- 0017  FORENSIC PATHOLOGISTS
                 340.67.007
>EXPAND
   *REF:- 0116  FOREIGN AID
                 341.232
   *REF:- 0065  FOREIGN MINISTERS CONFERENCES
                 341.182
   *REF:- 0017  FORENSIC PATHOLOGISTS
                 340.67.007
   *REF:- 0015  FORENSIC SCIENCE
                 340.6
   *REF:- 0118  FRANCO-RUSSIAN TALKS
                 341.232 (44)(470)
```

```
>RETAIN 0015
SYSTEM STRING 01 IS 340.6

ENTER COMMAND
>DICT
   01  T=TERM  O=SYST  ASM=0015  Q1=00  Q2=00  PDP=00
   340.6

ENTER COMMAND
>SEARCH
0002 DIRECT HITS
0019 ROOT HITS

ENTER COMMAND
>BROWSE
COMMENCE AT FIRST ROOT HIT:- 0023/01
  SCORE=100  UDC= 340.6:343.977
>N-5
  SCORE=100  UDC= 340.6
  SCORE=100  UDC= 340.6
  SCORE=040  UDC= 34.611:656.1.071(411 BELFAST)
  SCORE=040  UDC= 34.362.733
  SCORE=040  UDC= 34.17
>J 20
  SCORE=100  UDC= 340.6:614.84
>DICT
   01  T=SEAR  O=SYST  ASM=0015  Q1=00  Q2=00  PDP=01
   340.6

ENTER COMMAND
>PAD ALL
   01  RS=0  STF=0  HITS=0002
   02  RS=0  STF=1  HITS=0019

ENTER COMMAND
>TERM
ENTER TERM
>COD WAR
   *REF:- 0083  COD WAR WITH ICELAND
                341.225.1 (491.1)
>N 5
   *REF:- 0316  COLOURED RACES
                (=9)
   *REF:- 0201  COMMISSION ON INTERNMENT, NORTHERN IRELAND
                341.345 (411) 06.045 PARKER
   *REF:- 0200  COMMISSION ON NORTHERN IRELAND INTERNMENT
                341.345 (411) 06.045 PARKER
   *REF:- 0063  COMMONWEALTH CONFERENCE ON RHODESIA (R)
                341.181 (41-44) "1966" 341.231 U.D.I.(674)
   *REF:- 0062  COMMONWEALTH CONFERENCES
                341.181(41-44)
>RETAIN 0083
SYSTEM STRING 02 IS 341.225.1 (491.1)

ENTER COMMAND
>DICT
   02  T=TERM  O=SYST  ASM=0083  Q1=00  Q2=00  PDP=00
   341.225.1 (491.1)

ENTER COMMAND
>SEARCH
0186 DIRECT HITS
0086 ROOT HITS
```

```
ENTER COMMAND
>BROWSE
COMMENCE AT FIRST ROOT HIT:- 0081/12
  SCORE=100  UDC= 341.225.1(491.1)061.3
>N 3
  SCORE=100  UDC= 341.225.1(491.1)061.3(42)
  SCORE=100  UDC= 341.225.1(491.1)061.3(42)
  SCORE=100  UDC= 341.225.1(491.1)061.3(491.1 REYKJAVIK)
>DICT
   02  T=SEAR  O=SYST  ASM=0083  Q1=00  Q2=00  PDP=03
   341.225.1 (491.1)

ENTER COMMAND
>PAD
   03  RS=0  STF=0  HITS=0186

ENTER COMMAND
>PAD 4
   04  RS=0  STF=1  HITS=0086

ENTER COMMAND
>MYUDC
ENTER UDC STRING
>341

ENTER COMMAND
>RETAIN
USER STRING 03 IS 341

ENTER COMMAND
>SEARCH
0001 DIRECT HITS
3035 ROOT HITS

ENTER COMMAND
>DICT
   03  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=05
   341

ENTER COMMAND
>PAD
   05  RS=0  STF=0  HITS=0001

ENTER COMMAND
>PAD 6
   06  RS=0  STF=1  HITS=3035

ENTER COMMAND
>EDIT
-341
>   .1
341.1

ENTER COMMAND
>RETAIN
USER STRING 04 IS 341.1

ENTER COMMAND
>DICT
   04  T=TERM  O=USER  ASM=0000  Q1=00  Q2=00  PDP=00
   341.1
```

```
ENTER COMMAND
>SEARCH
0001 DIRECT HITS
0567 ROOT HITS

ENTER COMMAND
>BROWSE
COMMENCE AT FIRST ROOT HIT:- 0024/09
  SCORE=100  UDC= 341.1(494.51 GENEVA)
>J 20
  SCORE=100  UDC= 341.123
>N
  SCORE=100  UDC= 341.123
>N
  SCORE=100  UDC= 341.123
>BASE
  SCORE=100  UDC= 341.1(494.51 GENEVA)
>DICT ALL
   01  T=SEAR  O=SYST  ASM=0015  Q1=00  Q2=00  PDP=01
   340.6
   02  T=SEAR  O=SYST  ASM=0083  Q1=00  Q2=00  PDP=03
   341.225.1 (491.1)
   03  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=05
   341
   04  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=07
   341.1

ENTER COMMAND
>PAD ALL
   01  RS=0  STF=0  HITS=0002
   02  RS=0  STF=1  HITS=0019
   03  RS=0  STF=0  HITS=0186
   04  RS=0  STF=1  HITS=0086
   05  RS=0  STF=0  HITS=0001
   06  RS=0  STF=1  HITS=3035
   07  RS=0  STF=0  HITS=0001
   08  RS=0  STF=1  HITS=0567

ENTER COMMAND
>RESET 3
CURRENT DICT NO:- 03

ENTER COMMAND
>DICT
   03  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=05
   341

ENTER COMMAND
>PAD
   05  RS=0  STF=0  HITS=0001

ENTER COMMAND
>EXPAND 2
   *REF:- 0083  COD WAR WITH ICELAND
                341.225.1 (491.1)
>EXPAND
   *REF:- 0213  CLAUDIA GUN RUNNING INCIDENT,1973 (R)
                341.355.1 (412) 629.123.4 (431) CLAUDIA
   *REF:- 0289  COCKNEYS (*)
                (=20)(421.25/27)
   *REF:- 0083  COD WAR WITH ICELAND
                341.225.1 (491.1)
   *REF:- 0316  COLOURED RACES
                (=9)
   *REF:- 0201  COMMISSION ON INTERNMENT, NORTHERN IRELAND
                341.345 (411) 94.045 PAPER
```

```
>CURRENT
CURRENT DICT NO:- 03

ENTER COMMAND
>AFILE 0083 UP
   *REF:- 0083  COD WAR WITH ICELAND
                 341.225.1 (491.1)
UDC SUPERIOR
   *REF:- 0082  FISHING REGULATIONS
                 341.225.1
>
UDC SUPERIOR
   *REF:- 0067  INTERNATIONAL RELATIONS
                 341.2
>
UDC SUPERIOR
   *REF:- 0019  INTERNATIONAL LAW
                 341
>
UDC SUPERIOR
   *REF:- 0002  LAW
                 34
>
EMPTY STRING AND RETURN

ENTER COMMAND
>AFILE SEARCH
ENTER UDC STRING
>34
   *REF:- 0002  LAW
                 34
>AFILE 0002 DOWN
   *REF:- 0002  LAW
                 34
CD=01 CHAIN CONTINUES
340
CD=02 CHAIN CONTINUES
   *REF:- 0019  INTERNATIONAL LAW
                 341
CD=12 CHAIN CONTINUES
34.
SELECT TERM CD NUMBER
>2
CD=12 CHAIN CONTINUES
341.
SELECT TERM CD NUMBER
>12
CD=01 CHAIN CONTINUES
341.0
CD=02 CHAIN CONTINUES
   *REF:- 0022  WORLD ORGANIZATION
                 341.1
CD=03 CHAIN CONTINUES
   *REF:- 0067  INTERNATIONAL RELATIONS
                 341.2
CD=04 CHAIN CONTINUES
   *REF:- 0176  WAR, LAW OF
                 341.3
SELECT TERM CD NUMBER
>2
```

```
CD=03 CHAIN CONTINUES
   *REF:- 0023  WORLD GOVERNMENT ORGANIZATIONS
                341.12
CD=07 CHAIN CONTINUES
341.16
CD=08 CHAIN CONTINUES
   *REF:- 0053  STATES, COMMUNITIES OF
                341.17
CD=09 CHAIN CONTINUES
   *REF:- 0059  INTER-STATE CONFERENCES
                341.18
SELECT TERM CD NUMBER
>RESET 1
CURRENT DICT NO:- 01

ENTER COMMAND
>DICT
   01  T=SEAR  O=SYST  ASM=0015  Q1=00  Q2=00  PDP=01
   340.6

ENTER COMMAND
>AFILE 0015 ACROSS
   *REF:- 0015  FORENSIC SCIENCE
                340.6
>N 2
   *REF:- 0016  CRIME LABORATORIES (R)
                340.6 ; 5.001.5
   *REF:- 0018  PATHOLOGISTS, FORENSIC
                340.67.007
>RETAIN 0018
SYSTEM STRING 05 IS 340.67.007

ENTER COMMAND
>DICT ALL
   01  T=SEAR  O=SYST  ASM=0015  Q1=00  Q2=00  PDP=01
   340.6
   02  T=SEAR  O=SYST  ASM=0083  Q1=00  Q2=00  PDP=03
   341.225.1 (491.1)
   03  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=05
   341
   04  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=07
   341.1
   05  T=TERM  O=SYST  ASM=0018  Q1=00  Q2=00  PDP=00
   340.67.007

ENTER COMMAND
>SEARCH
0000 DIRECT HITS
0000 ROOT HITS

ENTER COMMAND
>PAD ALL
   01  RS=0  STF=0  HITS=0002
   02  RS=0  STF=1  HITS=0019
   03  RS=0  STF=0  HITS=0186
   04  RS=0  STF=1  HITS=0086
   05  RS=0  STF=0  HITS=0001
   06  RS=0  STF=1  HITS=3035
   07  RS=0  STF=0  HITS=0001
   08  RS=0  STF=1  HITS=0567
   09  RS=0  STF=0  HITS=0000
   10  RS=0  STF=1  HITS=0000
```

```
ENTER COMMAND
>EDIT
-340.67.007
>     #####
340.6

ENTER COMMAND
>AFILE SEARCH
ENTER UDC STRING
>
   *REF:- 0015  FORENSIC SCIENCE
                 340.6
>N 2
   *REF:- 0016  CRIME LABORATORIES (R)
                 340.6 : 5.001.5
   *REF:- 0018  PATHOLOGISTS, FORENSIC
                 340.67.007
>TIME
      16:03:13     DECEMBER 8, 1977

ENTER COMMAND
>MYUDC
ENTER UDC STRING
>.007

ENTER COMMAND
>RETAIN
USER STRING 06 IS .007

ENTER COMMAND
>CURRENT
CURRENT DICT NO:- 06

ENTER COMMAND
>DICT
   06  T=TERM  O=USER  ASM=0000  Q1=00  Q2=00  PDP=00
   .007

ENTER COMMAND
>THRESH
CURRENT THRESHOLD IS:- 100%

ENTER COMMAND
>QSEARCH BY 06 ON ALL
QSEARCH WITH THRESHOLD OF 100 ? (YES OR NO)
>YES
WHOLE UDC SAVED ON DICT - 07

ENTER COMMAND
>DICT ALL
   01  T=SEAR  O=SYST  ASM=0015  Q1=00  Q2=00  PDP=01
   340.6
   02  T=SEAR  O=SYST  ASM=0083  Q1=00  Q2=00  PDP=03
   341.225.1 (491.1)
   03  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=05
   341
   04  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=07
   341.1
   05  T=SEAR  O=SYST  ASM=0018  Q1=00  Q2=00  PDP=09
   340.67.007
   06  T=QSEA  O=USER  ASM=0000  Q1=07  Q2=07  PDP=12
   .007
   07  T=SAVE  O=UDC   ASM=0000  Q1=00  Q2=00  PDP=11
   .007
```

```
ENTER COMMAND
>PAD ALL
   01   RS=0   STF=0   HITS=0002
   02   RS=0   STF=1   HITS=0019
   03   RS=0   STF=0   HITS=0186
   04   RS=0   STF=1   HITS=0086
   05   RS=0   STF=0   HITS=0001
   06   RS=0   STF=1   HITS=3035
   07   RS=0   STF=0   HITS=0001
   08   RS=0   STF=1   HITS=0567
   09   RS=0   STF=0   HITS=0000
   10   RS=0   STF=1   HITS=0000
   11   RS=0   STF=1   HITS=5565
   12   RS=1   STF=1   HITS=0042

ENTER COMMAND
>RESULT
   0001   REL=100%   ABS=100%   [BBC] 654.19:62.007
   0002   REL=100%   ABS=100%   [BBC] 654.19:62.007
   0003   REL=100%   ABS=100%   [BRANNEN THERMOMETERS] 331.892:536.5.007.
 "1972"
   0004   REL=100%   ABS=100%   [CUBITTS] 331.892:69.007.25(421.5 THAMESM
D)"1969"
   0005   REL=100%   ABS=100%   [FINE TUBES] (422.4 PLYMOUTH)331.892 "197
 323.233.007
   0006   REL=100%   ABS=100%   [FINE TUBES] (422.4 PLYMOUTH)331.892 "197
 323.233.007
   0007   REL=100%   ABS=100%   [G.E.C.].007.33: 331.892 "1973" 331.4: 33
215
   0008   REL=100%   ABS=100%   [HEATON'S] 656.073.235:629.31.007.25:331.
 "1972"
   0009   REL=100%   ABS=100%   [KRUPPS] 725.4.007.25(431.)
   0010   REL=100%   ABS=100%   (=4) 69.007.25
   0011   REL=100%   ABS=100%   (=51) 725.4.007.25
   0012   REL=100%   ABS=100%   (=82) 380.8.007 (421.2)
   0013   REL=100%   ABS=100%   (=86) 725.4.007.25
   0014   REL=100%   ABS=100%   (=9) 351.82.007.25
   0015   REL=100%   ABS=100%   (=9) 656.05.007: 3-055.2(421.211)
   0016   REL=100%   ABS=100%   (=9)664.65.007.25(423.14 BIRMINGHAM)
   0017   REL=100%   ABS=100%   (=9)725.4.007.25
   0018   REL=100%   ABS=100%   (=924) 629.114.5.007:335.4 (5-011)
   0019   REL=100%   ABS=100%   (=924) 725.4.007.25 (569.4)
   0020   REL=100%   ABS=100%   (=927)321.21.007
   0021   REL=100%   ABS=100%   (=927)321.21.007: 725.171(569.5 AMMAN)
   0022   REL=100%   ABS=100%   (=927) 631.007.1 (286.263.1)
   0023   REL=100%   ABS=100%   (=927) 631.007.2
   0024   REL=100%   ABS=100%   (=927) 64.033: (534): 380.8.007:627.21: :
.33
   0025   REL=100%   ABS=100%   (=927) 796.4.007 (569.4)
   0026   REL=100%   ABS=100%   (=927 BEDOUINS)631.007.2 (532)
   0027   REL=100%   ABS=100%   M99- : 343.611: 32.007 (747 NEW YORK)
   0028   REL=100%   ABS=100%   34:323.233.007:351.741
   0029   REL=100%   ABS=100%   34.096:323.233.007:622.007.25(427.3 DUNFI
LINE)
   0030   REL=100%   ABS=100%   34.096:323.233.007:69.007.25(423.11SHREW
RY)
   0031   REL=100%   ABS=100%   34.096:323.233.007(411.5 ARMAGH)
   0032   REL=100%   ABS=100%   34.096:323.233.007(5-11)355.4
   0033   REL=100%   ABS=100%   340.6.616.89(473 MOSCOW)SERBSKY INSTUTUT.
.007(100-15)
```

```
    0034   REL=100%   ABS=100%   341.231 U.D.I.(674)321.21.007(=96)
    0035   REL=100%   ABS=100%   341.231 UDI (674)343.261 (674)07.007
    0036   REL=100%   ABS=100%   341.231 UDI (674)343.264:07.007
    0037   REL=100%   ABS=100%   341.231 UDI (674)663.97.05:631.007.1
    0038   REL=100%   ABS=100%   341.232.2: 322.1.007.1
    0039   REL=100%   ABS=100%   341.345.007 (41)
    0040   REL=100%   ABS=100%   341.358.007
    0041   REL=100%   ABS=100%   341.358.007
    0042   REL=100%   ABS=100%   341.358.007

ENTER COMMAND
>EDIT
-.007
>321#
321

ENTER COMMAND
>RETAIN
USER STRING 08 IS 321

ENTER COMMAND
>THRESH 66

ENTER COMMAND
>QSEARCH BY 08 ON 06
QSEARCH WITH THRESHOLD OF 066 ? (YES OR NO)
>YES
FACET CONSTRAINT ON 06? (YES OR NO)
>YES

ENTER COMMAND
>DICT
   08  T=QSEA  O=USER  ASM=0000  Q1=06  Q2=06  PDP=13
   321

ENTER COMMAND
>PAD
   13  RS=1  STF=1  HITS=0012

ENTER COMMAND
>RESULT
   0043   REL=066%   ABS=066%   DFINE TUBES] (422.4 PLYMOUTH)331.892 "
 323.233.007
   0044   REL=066%   ABS=066%   DFINE TUBES] (422.4 PLYMOUTH)331.892 "
 323.233.007
   0045   REL=100%   ABS=100%   (=927)321.21.007
   0046   REL=100%   ABS=100%   (=927)321.21.007: 725.171(569.5 AMMAN)
   0047   REL=066%   ABS=066%   M99 : 343.611: 32.007 (747 NEW YORK)
   0048   REL=066%   ABS=066%   34:323.233.007:351.741
   0049   REL=066%   ABS=066%   34.096:323.233.007:622.007.25(427.3 DU
LINE)
   0050   REL=066%   ABS=066%   34.096:323.233.007:69.007.25(423.11SHR
RY)
   0051   REL=066%   ABS=066%   34.096:323.233.007(411.5 ARMAGH)
   0052   REL=066%   ABS=066%   34.096:323.233.007(5-11)355.4
   0053   REL=100%   ABS=100%   341.231 U.D.I.(674)321.21.007(=96)
   0054   REL=066%   ABS=066%   341.232.2: 322.1.007.1
```

```
ENTER COMMAND
>THRESH 100

ENTER COMMAND
>RESULT
   0045  REL=100%  ABS=100%  (=927)321.21.007
   0046  REL=100%  ABS=100%  (=927)321.21.007: 725.171(569.5 AM
   0053  REL=100%  ABS=100%  341.231 U.D.I.(674)321.21.007(=96)

ENTER COMMAND
>RESET 1
CURRENT DICT NO:- 01

ENTER COMMAND
>BROWSE
COMMENCE AT FIRST ROOT HIT:- 0023/01
  SCORE=100  UDC= 340.6:343.977
>J 20
  SCORE=040  UDC= 341
>SAVE 10
SAVE SET STORED ON DICT NO:- 09
>CURRENT
CURRENT DICT NO:- 09

ENTER COMMAND
>DICT
   09  T=SAVE  O=UDC   ASM=0000  Q1=00  Q2=00  PDF=14
   340.6

ENTER COMMAND
>PAD
   14  RS=0  STF=1  HITS=0010

ENTER COMMAND
>MYUDC
ENTER UDC STRING
>(=1

ENTER COMMAND
>RETAIN
USER STRING 10 IS (=1

ENTER COMMAND
>DICT
   10  T=TERM  O=USER  ASM=0000  Q1=00  Q2=00  PDF=00
   (=1

ENTER COMMAND
>THRESH
CURRENT THRESHOLD IS:- 100%

ENTER COMMAND
>THRESH 66

ENTER COMMAND

>QSEARCH BY 10 ON 01 TO 05
QSEARCH WITH THRESHOLD OF 066 ? (YES OR NO)
>YES
FACET CONSTRAINT ON 01? (YES OR NO)
>YES
FACET CONSTRAINT ON 02? (YES OR NO)
>YES
FACET CONSTRAINT ON 03? (YES OR NO)
>YES
```

```
FACET CONSTRAINT ON 04? (YES OR NO)
>YES
FACET CONSTRAINT ON 05? (YES OR NO)
>YES
NO RELEVANT HITS FOR TERM - 05

ENTER COMMAND
>PAD
   15  RS=1  STF=1  HITS=0000

ENTER COMMAND
>QSEARCH BY 10 ON 01 TO 05
QSEARCH WITH THRESHOLD OF 066 ? (YES OR NO)
>YES
NEW DICT ENTRY NO:- 11
FACET CONSTRAINT ON 01? (YES OR NO)
>NO
FACET CONSTRAINT ON 02? (YES OR NO)
>NO
FACET CONSTRAINT ON 03? (YES OR NO)
>NO
FACET CONSTRAINT ON 04? (YES OR NO)
>NO
FACET CONSTRAINT ON 05? (YES OR NO)
>NO
NO RELEVANT HITS FOR TERM - 05

ENTER COMMAND
>PAD
   16  RS=1  STF=1  HITS=0009

ENTER COMMAND
>RESULT
   0055  REL=100%  ABS=100%  341.123:341.431(662)(=1.5)
   0056  REL=100%  ABS=100%  341.123.042:341.431(662)(=1.5)
   0057  REL=100%  ABS=100%  341.231(729.724.27(=1.729.724.2)(42
   0058  REL=100%  ABS=100%  341.234 (=1.541.4)(549.3)
   0059  REL=100%  ABS=100%  341.234 (=1.541.4)(549.3)
   0060  REL=100%  ABS=100%  341.234(=1.541.4)(549.3)
   0061  REL=100%  ABS=100%  341.346 (=1947)
   0062  REL=100%  ABS=100%  341.123:341.431(662)(=1.5)
   0063  REL=100%  ABS=100%  341.123.042:341.431(662)(=1.5)

ENTER COMMAND
>DICT ALL
   01  T=SEAR  O=SYST  ASM=0015  Q1=00  Q2=00  PDP=01
   340.6
   02  T=SEAR  O=SYST  ASM=0083  Q1=00  Q2=00  PDP=03
   341.225.1 (491.1)
   03  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=05
   341
   04  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=07
   341.1
   05  T=SEAR  O=SYST  ASM=0018  Q1=00  Q2=00  PDP=09
   340.67.007
   06  T=QSEA  O=USER  ASM=0000  Q1=07  Q2=07  PDP=12
   .007
   07  T=SAVE  O=UDC   ASM=0000  Q1=00  Q2=00  PDP=11
   .007
   08  T=QSEA  O=USER  ASM=0000  Q1=06  Q2=06  PDP=13
   321
   09  T=SAVE  O=UDC   ASM=0000  Q1=00  Q2=00  PDP=14
   340.6
   10  T=QSEA  O=USER  ASM=0000  Q1=01  Q2=05  PDP=15
   (=1
   11  T=QSEA  O=USER  ASM=0000  Q1=01  Q2=05  PDP=16
   (=1
```

```
ENTER COMMAND
>PAD ALL
   01  RS=0  STF=0  HITS=0002
   02  RS=0  STF=1  HITS=0019
   03  RS=0  STF=0  HITS=0186
   04  RS=0  STF=1  HITS=0086
   05  RS=0  STF=0  HITS=0001
   06  RS=0  STF=1  HITS=3035
   07  RS=0  STF=0  HITS=0001
   08  RS=0  STF=1  HITS=0567
   09  RS=0  STF=0  HITS=0000
   10  RS=0  STF=1  HITS=0000
   11  RS=0  STF=1  HITS=5565
   12  RS=1  STF=1  HITS=0042
   13  RS=1  STF=1  HITS=0012
   14  RS=0  STF=1  HITS=0010
   15  RS=1  STF=1  HITS=0000
   16  RS=1  STF=1  HITS=0009

ENTER COMMAND
>@
END OF EXECUTION
>@XQT FILMLIB.RETRIEVE

ENTER COMMAND
>MYUDC
ENTER UDC STRING
>62

ENTER COMMAND
>RETAIN
USER STRING 01 IS 62

ENTER COMMAND
>FSEARCH
0008 DIRECT HITS
0624 ROOT HITS

ENTER COMMAND
>BROWSE
COMMENCE AT FIRST ROOT HIT:- 0257/05
  SCORE=100  UDC= [BBC] 654.19:62.007
>N 5
  SCORE=100  UDC= [BBC] 654.19:62.007
  SCORE=100  UDC= M99 : 656.1.08 : 620.16 : 621.381.95
  SCORE=100  UDC= 341.24: 621.039 (47)(73)"1973"
  SCORE=100  UDC= 34.096(427.3DUNFERMLINE)323.22(427.3LONGANNET)62
2:622.
  SCORE=100  UDC= 34.096(427.3DUNFERMLINE)323.22(427.3LONGANNET)62
2:622.
>MYUDC
ENTER UDC STRING
>341.123

ENTER COMMAND
>RETAIN
USER STRING 02 IS 341.123

ENTER COMMAND
>SEARCH
0020 DIRECT HITS
0324 ROOT HITS
```

```
ENTER COMMAND
>EDIT
-341.123
>.007###
.007

ENTER COMMAND
>RETAIN
USER STRING 03 IS .007

ENTER COMMAND
>THRESH
CURRENT THRESHOLD IS:- 100%

ENTER COMMAND
>QSEARCH BY 03 ON 01 TO 02
QSEARCH WITH THRESHOLD OF 100 ? (YES OR NO)
>YES
FACET CONSTRAINT ON 01? (YES OR NO)
>YES

FACET CONSTRAINT ON 02? (YES OR NO)
>NO

ENTER COMMAND
>PAD
   05  RS=1  STF=1  HITS=0005

ENTER COMMAND
>RESULT
   0001  REL=100%  ABS=100%  [BBC] 654.19:62.007
   0002  REL=100%  ABS=100%  [BBC] 654.19:62.007
   0003  REL=100%  ABS=100%  34.096:323.233.007:622.007.25(427.3 )
LINE)
   0004  REL=100%  ABS=100%  (=924) 629.114.5.007:335.4 (5-011).
   0005  REL=100%  ABS=100%  [HEATON'S] 656.073.235:629.31.007.25
 "1972"

ENTER COMMAND
>RESET 2
CURRENT DICT NO:- 02

ENTER COMMAND
>FSEARCH
NEW DICT ENTRY NO:- 04
0232 DIRECT HITS
0151 ROOT HITS

ENTER COMMAND

>DICT ALL
   01  T=FSEA  O=USER  ASM=0000  Q1=00  Q2=00  PDP=01
   62
   02  T=SEAR  O=USER  ASM=0000  Q1=00  Q2=00  PDP=03
   341.123
   03  T=QSEA  O=USER  ASM=0000  Q1=01  Q2=02  PDP=05
   .007
   04  T=FSEA  O=USER  ASM=0000  Q1=00  Q2=00  PDP=06
   341.123
```

```
ENTER COMMAND
>PAD ALL
   01  RS=0  STF=1  HITS=0008
   02  RS=0  STF=1  HITS=0624
   03  RS=0  STF=0  HITS=0020
   04  RS=0  STF=1  HITS=0326
   05  RS=1  STF=1  HITS=0005
   06  RS=0  STF=1  HITS=0232
   07  RS=0  STF=1  HITS=0151

ENTER COMMAND
>QSEARCH BY 01 ON ALL
QSEARCH WITH THRESHOLD OF 100 ? (YES OR NO)
>YES
WHOLE UDC SAVED ON DICT - 05
NEW DICT ENTRY NO:- 06


ENTER COMMAND
>PAD
   09  RS=1  STF=1  HITS=0508

ENTER COMMAND
>DICT
   06  T=QSEA  O=USER  ASM=0000  Q1=05  Q2=05  PDP=09
   62

ENTER COMMAND
>@
END OF EXECUTION
>@XQT FILMLIB.RETRIEVE

ENTER COMMAND
>MYUDC
ENTER UDC STRING
>DAN AIR

ENTER COMMAND
>RETAIN
USER STRING 01 IS DAN AIR

ENTER COMMAND
>DICT ALL
   01  T=TERM  O=USER  ASM=0000  Q1=00  Q2=00  PDP=00
   DAN AIR

ENTER COMMAND
>PAD ALL

ENTER COMMAND
>QSEARCH BY 01 ON ALL
QSEARCH WITH THRESHOLD OF 100 ? (YES OR NO)
>YES
WHOLE UDC SAVED ON DICT - 02


ENTER COMMAND
>DICT ALL
   01  T=QSEA  O=USER  ASM=0000  Q1=02  Q2=02  PDP=02
   DAN AIR
   02  T=SAVE  O=UDC   ASM=0000  Q1=00  Q2=00  PDP=01
   DAN AIR
```

```
ENTER COMMAND
>PAD ALL
   01  RS=0  STF=1  HITS=5565
   02  RS=1  STF=1  HITS=0001

ENTER COMMAND
>RESULT
   0001  REL=100%  ABS=100%  341.226(661)656.7.07 DAN-AIR:629.138.5 )
NG 707

ENTER COMMAND
>TERM
ENTER TERM
>TERRITORY
   *REF:- 0070  TERRITORY (DISPUTED)
                341.221
>AFILE 0070 DA
   *REF:- 0070  TERRITORY (DISPUTED)
                341.221
CD=12 CHAIN CONTINUES
341.221.
CD=18 CHAIN CONTINUES
341.221(
SELECT TERM CD NUMBER
>18
CD=06 CHAIN CONTINUES
341.221(5
SELECT TERM CD NUMBER
>6
CD=11 CHAIN CONTINUES
341.221(5-
SELECT TERM CD NUMBER
>11
CD=01 CHAIN CONTINUES
341.221(5-0
SELECT TERM CD NUMBER
>1
CD=02 CHAIN CONTINUES
341.221(5-01
SELECT TERM CD NUMBER
>2
CD=02 CHAIN CONTINUES
   *REF:- 0074  "OCCUPIED TERRITORIES" (MID-EAST PROBLEM)
                341.221 (5-011)
SELECT TERM CD NUMBER
>RETAIN 0074
SYSTEM STRING 03 IS 341.221 (5-011)

ENTER COMMAND
>TERM
ENTER TERM
>PARKER
   *REF:- 0199  PARKER COMMISSION ON NORTHERN IRELAND INTERNMENT
                341.345 (411) 06.045 PARKER
>@
END OF EXECUTION
```

In the example below, the user begins to print 99 UDC entries but then decides to stop the printing and go on to do something else. This involves use of the BREAK key and the '@@X 0' command.

```
>@XQT FILMLIB.RETRIEVE

ENTER COMMAND
>MYUDC
ENTER UDC STRING
>34

ENTER COMMAND
>RETAIN
USER STRING 01 IS 34

ENTER COMMAND
>SEARCH
0004 DIRECT HITS
3342 ROOT HITS

ENTER COMMAND
>BROWSE
COMMENCE AT FIRST ROOT HIT:- 0003/05
  SCORE=100  UDC= 34:061 BAR COUNCIL
>N 99
  SCORE=100  UDC= 34:061.2 JUSTICE:347.426.1:656.1.08.
  SCORE=100  UDC= 34:323.233.007:351.741
  SCORE=100  UDC= 34:337.91 EEC
  SCORE=100  UDC= 34:337.91 EEC(42)"1973-"
  SCORE=100  UDC= 34:341.181(41-44)
  SCORE=100  UDC= 34:  SCORE=100  UDC= 34:
                          *****BREAK key pressed here
*OUTPUT INTERRUPT*
@@X 0
  SCORE=100  UDC= 34:347.254
>TIME
      16:31:06     DECEMBER 8, 1977

ENTER COMMAND
```

In the following example, the user initiates a very long QSEARCH, but then decides to stop it. This involves terminating retrieval program execution by means of the '@@X T' command. Having done this, the user stops the run (by @FIN), breaks the connection to Milton Keynes (@@TERM) and hangs up the phone.

(NOTE the unwillingness of the computer to respond when it's busy - pressing RETURN whilst the QSEARCH is underway results in '*WAIT' messages).

```
>@XQT FILMLIB.RETRIEVE

ENTER COMMAND
>MYUDC
ENTER UDC STRING
>34

ENTER COMMAND
>RETAIN
USER STRING 01 IS 34

ENTER COMMAND
>THRESH 0

ENTER COMMAND
>QSEARCH BY 01 ON ALL
QSEARCH WITH THRESHOLD OF 000 ? (YES OR NO)
>YES
WHOLE UDC SAVED ON DICT - 02
                          .......RETURN pressed
*WAIT-LAST INPUT IGNORED*
                          .......RETURN pressed
*WAIT-LAST INPUT IGNORED*
@@X T
*EXECUTION TERMINATED*
RUN E'ED OFF ABORT ADR: 011022  BDI:000004
MORE INFO?>
>@FIN


RUNID: ROSTFF    ACCT: SCS              PROJECT: BSTAFF
    ROSTFF  ABORT
TIME:     TOTAL: 00:09:55.071       CBSUPS:  098545301
          CPU:   00:03:30.037       I/O:  00:05:10.956
          CC/ER: 00:01:14.077       WAIT: 00:25:29.568

SUAS USED:  $    36.98   SUAS REMAINING:  $  9779.78
IMAGES  READ:   249        PAGES:  39
START:    15:49:28 DEC 08,1977    FIN:  16:31:11 DEC 08,1977
*TERMINAL INACTIVE*
>@@TERM
```

A P P E N D I X 2

USERS GUIDE TO THE COMPUTERIZED BBC FILM LIBRARY UPDATE PROGRAM

Version 1 - December 1977

Contents

1. OPERATING INSTRUCTIONS

Introduction

This document is purely concerned with the "UPDATE" program used to maintain files referenced by "RETRIEVE" (which operates following the @XQT FILMLIB.RETRIEVE command), and it does not contain elementary details of signing-on, signing-off etc., which are contained exclusively in the documentation describing the RETRIEVE program (see - "Users Guide to the Computerized BBC Film Library Information Retrieval System").

Using UPDATE, the computer-based versions of the Subject Index, the Authority File and the main UDC file can be altered by insertion or deletion. When the files are assigned, the program is initiated by sending:

@XQT FILMLIB.UPDATE

and is terminated by sending a single @.

Using the UPDATE program

Having signed-on, and sent both the @RUN and @ADD commands, you can operate either RETRIEVE or UPDATE. Between @RUN and @FIN, the @ADD command should appear once only, whereas either of the @XQT commands (RETRIEVE or UPDATE) can appear any number of times and in any order. For example, the following would be valid:

Sign-on
@RUN .....
@ADD .....
@XQT FILMLIB.RETRIEVE
@XQT FILMLIB.UPDATE

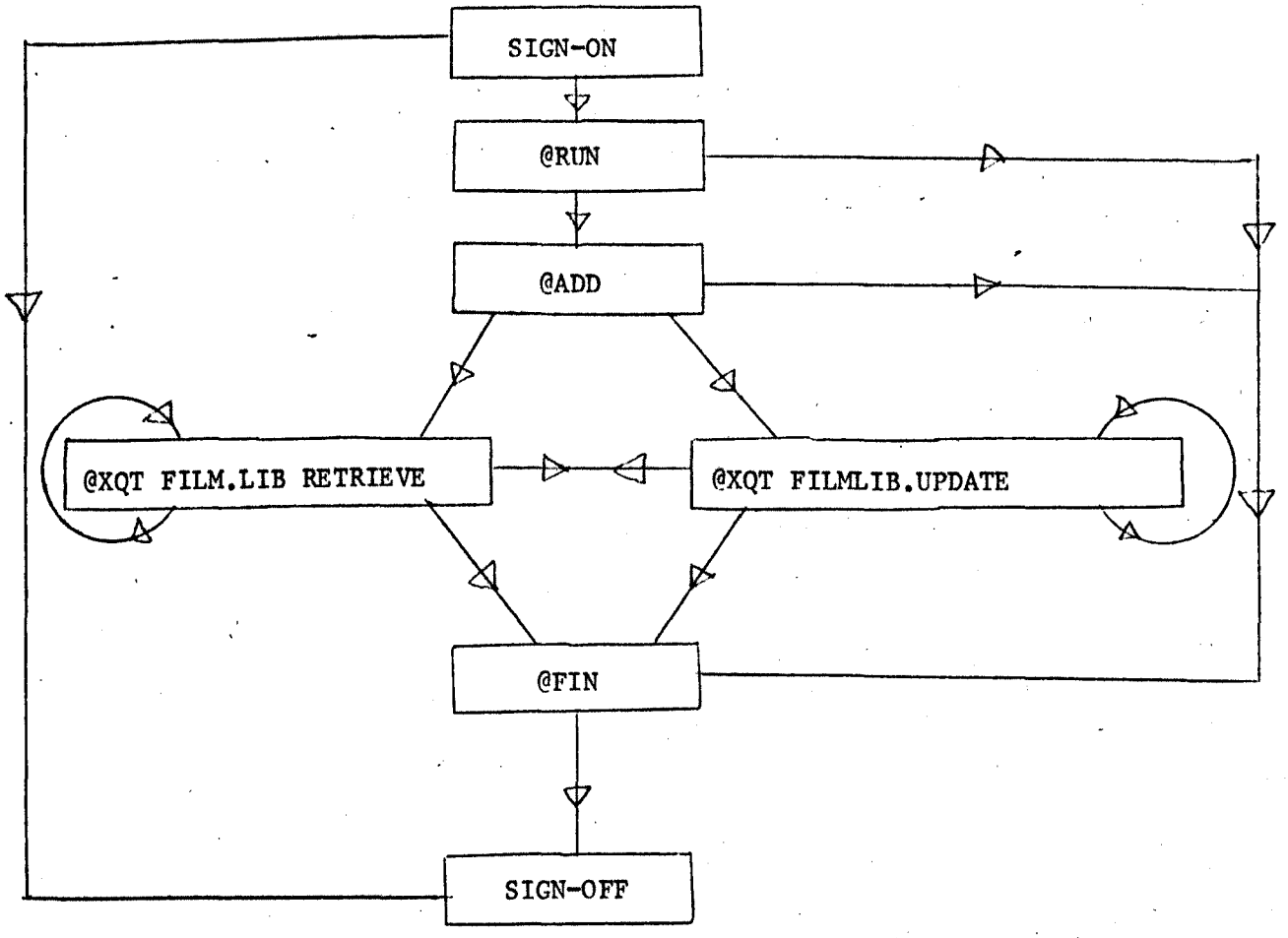
@XQT FILMLIB.UPDATE

@XQT FILMLIB.RETRIEVE

@FIN

Sign-off

or, in terms of a flowchart where arrows represent the only valid consecutive steps:



Having typed @XQT FILMLIB.UPDATE, the UPDATE program begins to operate and you will be asked:

ARE THE FILES OPEN?

You should reply by typing YES. If the computer fails to receive a YES reply, it will go on to print ENTER PASSWORD, and if you get this message you should re-type the @XQT command, so starting again from the beginning.

Next, the computer will print:

ENTER MODE, UDC or AS

If you wish to update the UDC file you should reply by typing UDC. The computer views the Subject Index and the Authority File as one entity, so if you wish to update this file you should type AS (= Authority/Subject). Either way, the computer will respond by printing.

ENTER COMMAND

Now you can go on to ammend whichever file you have chosen. [If at any stage you want to change files - UDC to AS, or AS to UDC - simply type MODE in response to the ENTER COMMAND message, and the computer will ask for your new choice of file.]

UPDATE commands

Type the commands as underlined.

INSERT (or IN)

Action: Make a new entry in the file being updated.

(i) If MODE = UDC.

The computer responds by printing:

ENTER NEW UDC KEY

Whatever you type in response to this message will be interpreted as a new candidate for entry in the UDC file. Before actually

making the insertion however, you will be given the chance to change the first number you submitted, eg:

```
ENTER NEW UDC KEY
>636.8
TYPE NO TO CORRECT:- 636.8
>
```

If you meant to type 636.9, then replying with a NO at this stage will result in a repeat of the ENTER NEW UDC KEY message, thus enabling you to correct your earlier error. This safety device may also be used to overcome telephone interference.

If you're happy with your number (636.8 above) then you should simply press RETURN, and it will be inserted in the UDC file. The ENTER COMMAND message will result when this has been done. In the following example, user input is characterized by having the > symbol in the first character position:

```
ENTER COMMAND
>INSERT
ENTER NEW UDC KEY
>634.3
TYPE NO TO CORRECT:- 634.3
>
ENTER COMMAND
>INSERT
ENTER NEW UDC KEY
>634.7
TYPE NO TO CORRECT:- 634.7
>NO
ENTER NEW UDC KEY
>634.8
TYPE NO TO CORRECT:- 634.8
>
ENTER COMMAND
```

(ii) If MODE = AS.

For making an insertion in the Authority/Subject file, the computer needs both a UDC number and a Subject literal. Try

following an example:

```
ENTER COMMAND
>INSERT
ENTER NEW UDC AUTHORITY STRING
>711.2
TYPE NO TO CORRECT:- 711.2
>
ENTER NEW SUBJECT
>APPLES
TYPE NO TO CORRECT:- APPLES
>
ENTER COMMAND
>INSERT
ENTER NEW UDC AUTHORITY STRING
>721.4
TYPE NO TO CORRECT:- 721.4
>NO
ENTER NEW UDC AUTHORITY STRING
>736.5
TYPE NO TO CORRECT:- 736.5
>
ENTER NEW SUBJECT
>BATTLES
TYPE NO TO CORRECT:- BATTLES
>NO
ENTER NEW SUBJECT
>BOTTLES
TYPE NO TO CORRECT:- BOTTLES
>
ENTER COMMAND
```

This results in two new entries being made in both the Authority File and the Subject Index:

711.2	and	736.5
APPLES		BOTTLES

MINSERT (or MI)

Action: Make any number of new entries in the file being updated.

(i) If MODE = UDC

The computer continues to ask for new entries until you tell

it to stop by typing $END, eg:-

```
ENTER COMMAND
>MINSERT
ENTER NEW UDC KEY
>811.1
TYPE NO TO CORRECT:- 811.1
>
ENTER NEW UDC KEY
>811.2
TYPE NO TO CORRECT:- 811.2
>
ENTER NEW UDC KEY
>811.W
TYPE NO TO CORRECT:- 811.W
>NO
ENTER NEW UDC KEY
>811.3
TYPE NO TO CORRECT:- 811.3
>
ENTER NEW UDC KEY
>811.4
TYPE NO TO CORRECT:- 811.4
>
ENTER NEW UDC KEY
>$END
ENTER COMMAND
```

(ii) If MODE = AS

Not only can you make any number of entries involving a UDC number/Subject literal pair, but you can attach more than one Subject literal to just one UDC number. Again, you must type $END to stop input, eg:-

```
ENTER COMMAND
>MINSERT
ENTER NEW UDC AUTHORITY STRING
>844.1
TYPE NO TO CORRECT:- 844.1
>
ENTER NEW SUBJECT
>CARS
TYPE NO TO CORRECT:- CARS
>
CONTINUE A-S INPUT
>844.2
TYPE NO TO CORRECT:- 844.2
>
ENTER NEW SUBJECT
>DOGS
TYPE NO TO CORRECT:- DOGS
>
CONTINUE A-S INPUT
>CURS
TYPE NO TO CORRECT:- CURS
>
CONTINUE A-S INPUT
>PUPS
TYPE NO TO CORRECT:- PUPS
>
CONTINUE A-S INPUT
>844.3
TYPE NO TO CORRECT:- 844.3
>
ENTER NEW SUBJECT
>GUNS
TYPE NO TO CORRECT:- GUNS
>
CONTINUE A-S INPUT
>$END
ENTER COMMAND
```

This results in the following new entries being made in both the Authority File and the Subject Index:

844.1	844.2	844.3
CARS	DOGS	GUNS
	844.2	
	CURS	
	844.2	
	PUPS	

When making multiple insertions in this way, you must obey the following rules:

(1) When a UDC number is requested, you must supply a UDC number.

(2) When a Subject literal is requested, you must supply a Subject literal.

(3) When the CONTINUE A-S INPUT message is printed, you can:-

(a) supply a further Subject literal to be attached to the most recent UDC authority number (i.e. add a synonym), or

(b) supply a new UDC authority number, or

(c) type $END

(4) You should never type $END in response to the ENTER NEW SUBJECT message.

DELETE (or DE)

Action: Delete an existing entry from the file being updated.

(i) If MODE = UDC

The computer responds by printing:

ENTER KEY

You should reply by typing a UDC number that already exists on the UDC file. If the computer can find no match for the key you type, it responds by printing:

NO EXACT MATCH FOR DELETE KEY

If an exact match is found however, the computer displays the match and then prints:

DELETE OR SCAN?

If you wish to delete the key, you simply type DELETE (or DE) a second time, and the computer prints the ENTER COMMAND message, signifying that the said key has been deleted. If, on the other hand, you want to look at adjacent keys on the UDC file, you should type SCAN (or SC), and you can then browse around the main UDC file using the following two commands:

N - show me the next entry

P - show me the previous entry

If you find a key that you want to delete whilst browsing in this way, you simply type DELETE, and the most recently displayed entry is deleted. Alternatively you can type another command by way of deciding against deletion. The reason for SCANing, is that the UDC file may contain several matches to your original key, and only by looking at them individually (by use of N and P) can you decide which one you want to delete.

Here's an example:

```
ENTER COMMAND
>DELETE
ENTER KEY
>340.796
P=0310  NM=340.96:341.48
NO EXACT MATCH FOR DELETE KEY
ENTER COMMAND
>DELETE
ENTER KEY
>341.123
P=0351  NM=341.123
DELETE OR SCAN
>DE
ENTER COMMAND
>DE
ENTER KEY
>34
P=0005  NM=34
DELETE OR SCAN
>SC
P=0005 KEY=34
>N
P=0004 KEY=34
>DE
ENTER COMMAND
>DE
ENTER KEY
>341
P=0311  NM=341
DELETE OR SCAN
>SCAN
P=0311 KEY=341
>N
P=0312 KEY=341(492.6 THE HAGUE,INTERNATIONAL COURT)323.1
>N
P=0313 KEY=341.1
>N
P=0314 KEY=341.1(494.51 GENEVA)
>N
P=0323 KEY=341.121
>P
P=0314 KEY=341.1(494.51 GENEVA)
>INSERT
ENTER NEW UDC KEY
```

(ii) If MODE = AS

Really, this is exactly the same as when MODE = UDC, except now, because file entries exist in pairs (UDC/Subject literal), you can specify the delete key either by UDC number or by Subject literal. For example, supposing you have an entry:

844.1

CARS

either of the following sequences serves to delete it:

```
ENTER COMMAND
>DELETE
ENTER KEY
>844.1
P=0356   NM=844.1
P=0356   NM=CARS
DELETE OR SCAN
>DELETE
ENTER COMMAND
```

```
ENTER COMMAND
>DELETE
ENTER KEY
>CARS
P=0356   NM=844.1
P=0356   NM=CARS
DELETE OR SCAN
>DELETE
ENTER COMMAND
```

Since a single UDC authority number can have multiple Subject literals associated with it (synonyms), the DELETE-SCAN-DELETE sequence can be used to find the exact UDC/Subject pair to be deleted, eg:-

```
>DE
ENTER KEY
>844.2
P=0359   NM=844.2
P=0359   NM=PUPS
DELETE OR SCAN
>SC
P=0359 KEY=844.2
P=0359 KEY=PUPS
>N
P=0358 KEY=844.2
P=0358 KEY=CURS
>DE
ENTER COMMAND
```

The same thing could have been achieved as follows:

```
ENTER COMMAND
>DE
ENTER KEY
>CURS
P=0358  NM=844.2
P=0358  NM=CURS
DELETE OR SCAN
>DE
ENTER COMMAND
```

RETRIEVE (or RE)

Action: Find the nearest match to a given key in the file being updated.

(i) If MODE = UDC

The computer responds by printing:

ENTER KEY

You should reply by giving it a UDC number, and it will then find the nearest match in the UDC file and display it.

(ii) If MODE = AS

Now you can supply either a UDC number or a Subject literal following the ENTER KEY message, thereby causing the nearest match in the Authority/Subject Index to be displayed.

SCAN (or SC) - following immediately after RETRIEVE.

Action: Permit browsing in the file being updated using commands:

N - show me the next entry

P - show me the previous entry

When MODE = AS, if the preceding RETRIEVE was via a UDC authority number, then browsing proceeds according to the ordering of the UDC entries, whereas if the RETRIEVE was via a Subject literal, browsing proceeds according to the alphabetical ordering of the Subject entries.

MODE (or MO)

Action: Change the file being updated.

See "Introduction".

The following example illustrates the use of RETRIEVE, SCAN and MODE commands:

```
>RE
ENTER KEY
>340.96
P=0019  NM=341
P=0019  NM=INTERNATIONAL LAW
ENTER COMMAND .
>SCAN
P=0019 KEY=341
P=0019 KEY=INTERNATIONAL LAW
>N
P=0020 KEY=341.012
P=0020 KEY=SELF-DETERMINATION
>N
P=0022 KEY=341.1
P=0022 KEY=WORLD ORGANIZATION
>MODE
ENTER MODE, UDC OR AS
>UD


ENTER COMMAND
>RE
ENTER KEY
>(=924)
P=4539  NM=(=924)
ENTER COMMAND
>SCAN
P=4539 KEY=(=924)
>N
P=4538 KEY=(=924)
>N
P=4537 KEY=(=924)
>N
P=4536 KEY=(=924)
```

SAVE (or SA)

The files that you amend during operation of the UPDATE program are merely copies of the main files. This is to guard against corruption of the files either by an inexperienced user, or by some computer fault. (In fact it is the @ADD FILMLIB.GO command that makes these temporary copies). If however, you are authorized to make ammendments to the main files themselves, you will want to ensure that the corrections made to the temporary files are subsequently applied to the main files. To do this you simply type:

SAVE

and the computer will respond by printing:

ENTER PASSWORD

If you type the wrong password in reply the computer will ask you for it again, but if it is correct the computer will terminate execution of UPDATE and set about updating the main files, during the course of which the following sequence of messages will result:

END OF EXECUTION
6 "READY" messages
1 "FURPUR" message
6 "BLOCKS COPIED" messages
>

If you don't receive this sequence of replies, you should contact me (0908 63188 - Brian Staff) before making any further use of the UPDATE program! If all goes well however, you can then either re-execute UPDATE, execute RETRIEVE, or sign-off. An example follows:

```
ENTER COMMAND
>SAVE
ENTER PASSWORD
>     ..............correct password was typed here
END OF EXECUTION
READY
READY
READY
READY
READY
READY
FURPUR27R2-4 72-8.1 12/08/77 15:52:05
    175 BLOCKS COPIED.
    265 BLOCKS COPIED.
     17 BLOCKS COPIED.
     17 BLOCKS COPIED.
     17 BLOCKS COPIED.
    109 BLOCKS COPIED.
READY
READY
READY
READY
READY
READY
>@XQT FILMLIB.RETRIEVE

ENTER COMMAND
```

Note!

During the course of a run (i.e. between @RUN and @FIN commands) only one set of temporary files is copied (i.e. only one @ADD command is issued). The SAVE command saves all the ammendments you have made in a single run prior to the issue of the SAVE, even though they might have been made in separate executions of UPDATE, eg:

```
> @RUN
> @ADD
> @XQT FILMLIB.UPDATE
.
. Updates (1)
.
.
> @
  END OF EXECUTION <------ Temporary files now contain
                           Updates (1), but not the main files.
```

```
> @XQT FILMLIB.UPDATE

.
. Updates (2)
.
.
> SAVE                    <------ Both sets of Updates, (1) and (2),
  END OF EXECUTION                are now applied to the main files.
  READY etc
```

This means that if you want to scrap all of your previous updates (without saving them) and then make some more updates (that you do want to save), you have to start a new run, eg:-

```
> @XQT FILMLIB.UPDATE

.
. Updates (1)
.
.
  ENTER COMMAND
> @FIN                    <------ Updates (1) are lost.
> @RUN ...
> @ADD ...
> @XQT FILMLIB.UPDATE

.
. Updates (2)             <------ Changes to be permanent
.
.
  SAVE                    <------ Updates (2) are applied to main files
>
```

Stopping the UPDATE program

To stop UPDATE without saving your amendments, simply type @ followed by RETURN. This will result in the END OF EXECUTION message and you can then either re-execute UPDATE, execute RETRIEVE or sign-off.

Errors and Abnormal Termination of UPDATE

The most common error that you encounter will have nothing to do with you, but will be brought about purely by a poor phone connection; so if you send a correct command, but receive strange responses - such as ILLEGAL COMMAND - you should re-type the command and hope for the best. Full details of line correction, computer breakdowns etc. are contained in the documentation of the RETRIEVE program, but I'll list a few last ditch measures below; for example, if:-

(i) all computer response ceases, or
(ii) a *WAIT LAST INPUT IGNORED* message greets your every command

First, you should hold down the CONTROL key and press the X key once. If nothing useful happens, press the RETURN key repeatedly. If this fails, type a single @ and press RETURN. If even this fails to provoke a sensible response, you should type:

@@X TIO followed by RETURN

which should result in the *EXECUTION TERMINATED* message. This must be followed by:

@FIN and RETURN

to ensure that possible file corruption is averted. To re-execute the program at this stage you will need to type:

@RUN .... etc
@ADD .... etc
@XQT .... etc

If the computer ever starts producing reems of unwanted output, you should use the following sequence:

press the BREAK key

send @@X 0 (0 = the letter)

If ever in doubt, contact Brian Staff (0908 63188)

If you receive any unexpected messages that you can't deal with - contact me. Particularly if you intend to SAVE your amendments, you must be wary of unexpected messages, and if you receive a message indicating a program error (rather than a typing error on your part) you should not SAVE your ammendments, but rather type @FIN and start again.

2. COMPLETE EXAMPLES

```
SCICON/OPEN UNIV. FRONT-END SYSTEM  8-DEC-77  10:21   LINE - 10,0
UNIVAC  SITE I/D: UHI207
*UNIVAC 1100 OPERATING SYSTEM  VER. 33R2-0U16   (RSI)*
>@RUN ROSTFF,SCS/BS,BSTAFF
DATE: 120877      TIME: 102134
>@ADD FILMLIB.GO
READY
READY
READY
READY
READY
READY
READY
FURPUR27R2-4 72-8.1 12/08/77 09:37:38
   175 BLOCKS COPIED.
   265 BLOCKS COPIED.
    17 BLOCKS COPIED.
    17 BLOCKS COPIED.
    17 BLOCKS COPIED.
   109 BLOCKS COPIED.
READY
READY
READY
READY
READY
READY
>@XQT FILMLIB.UPDATE
ARE THE FILES OPEN?
>YES
ENTER MODE, UDC OR AS
>UDC
ENTER COMMAND
>INSERT
ENTER NEW UDC KEY
>636,9
TYPE NO TO CORRECT:- 636.9
>
```

```
ENTER COMMAND
>SAVE
ENTER PASSWORD
>     ......................the correct password was entered here
END OF EXECUTION
READY
READY
READY
READY
READY
READY
FURPUR27R2-4 72-8.1 12/08/77 09:16:14
   175 BLOCKS COPIED.
   265 BLOCKS COPIED.
    17 BLOCKS COPIED.
    17 BLOCKS COPIED.
    17 BLOCKS COPIED.
   109 BLOCKS COPIED.
READY
READY
READY
READY
READY
READY
```

```
>@XQT FILMLIB.UPDATE
ARE THE FILES OPEN?
>YES
ENTER MODE, UDC OR AS
>UDC
ENTER COMMAND
>RETRIEVE
ENTER KEY
>341.12
P=0323  NM=341.121
ENTER COMMAND
>INSERT
ENTER NEW UDC KEY
>340.77
TYPE NO TO CORRECT:- 340.77
>
ENTER COMMAND
>RETRIEVE
ENTER KEY
>340.77
P=5567  NM=340.77
ENTER COMMAND
>SCAN
P=5567 KEY=340.77
>N
P=0310 KEY=340.96:341.48
>N
P=0311 KEY=341
>MINSERT
```

```
ENTER NEW UDC KEY
>340.90
TYPE NO TO CORRECT:- 340.90
>NO
ENTER NEW UDC KEY
>340.91
TYPE NO TO CORRECT:- 340.91
>
ENTER NEW UDC KEY
>340.92
TYPE NO TO CORRECT:- 340.92
>
ENTER NEW UDC KEY
>340.93
TYPE NO TO CORRECT:- 340.93
>
ENTER NEW UDC KEY
>$END
ENTER COMMAND
>RETRIEVE
ENTER KEY
>340.92
P=5569  NM=340.92
ENTER COMMAND
>SCAN
P=5569 KEY=340.92
>N
P=5570 KEY=340.93
>P
P=5569 KEY=340.92
>P
P=5568 KEY=340.91
>P
P=5567 KEY=340.77
>DELETE
ENTER KEY
>340.96
P=0310  NM=340.96:341.48
NO EXACT MATCH FOR DELETE KEY
ENTER COMMAND
>DELETE
ENTER KEY
>34
P=0005  NM=34
DELETE OR SCAN
>SCAN
P=0005 KEY=34
>N
P=0004 KEY=34
>P
P=0005 KEY=34
>DELETE
ENTER COMMAND
>MODE
ENTER MODE, UDC OR AS
>AS
ENTER COMMAND
```

```
>RE
ENTER KEY
>AIR TO GROUND
P=0346  NM=R13
P=0346  NM=AIR TO GROUND SHOTS (*)
ENTER COMMAND
>SCAN
P=0346 KEY=AIR TO GROUND SHOTS (*)
P=0346 KEY=R13
>N
P=0260 KEY=ALBERTSLUND
P=0260 KEY=(489 COPENHAGEN, ALBERTSLUND)
>N
P=0100 KEY=ALGERIA INDEPENDENCE PROBLEMS
P=0100 KEY=341.231(612)
>RE
ENTER KEY
>R13
P=0346  NM=R13
P=0346  NM=AIR TO GROUND SHOTS (*)
ENTER COMMAND
>SCAN
P=0346 KEY=R13
P=0346 KEY=AIR TO GROUND SHOTS (*)
>N
P=0345 KEY=R13
P=0345 KEY=AERIAL SHOTS (*)
>INSERT
ENTER NEW UDC AUTHORITY STRING
>101
TYPE NO TO CORRECT:- 101
>
ENTER NEW SUBJECT
>APPLES
TYPE NO TO CORRECT:- APPLES
>
ENTER COMMAND
>RE
ENTER KEY
>APPLES
P=0356  NM=101
P=0356  NM=APPLES
ENTER COMMAND
>MINSERT
ENTER NEW UDC AUTHORITY STRING
>102
TYPE NO TO CORRECT:- 102
>
ENTER NEW SUBJECT
>BUGS
TYPE NO TO CORRECT:- BUGS
>NO
ENTER NEW SUBJECT
>BAGS
TYPE NO TO CORRECT:- BAGS
>
CONTINUE A-S INPUT
```

```
>CARRIER BAGS
TYPE NO TO CORRECT:- CARRIER BAGS
>
CONTINUE A-S INPUT
>103
TYPE NO TO CORRECT:- 103
>
ENTER NEW SUBJECT
>CARS
TYPE NO TO CORRECT:- CARS
>
CONTINUE A-S INPUT
>104
TYPE NO TO CORRECT:- 104
>
ENTER NEW SUBJECT
>DOGS
TYPE NO TO CORRECT:- DOGS
>
CONTINUE A-S INPUT
>$END
ENTER COMMAND
>RE
ENTER KEY
>103
P=0359  NM=103
P=0359  NM=CARS
ENTER COMMAND
>SC
P=0359 KEY=103
P=0359 KEY=CARS
>N
P=0360 KEY=104
P=0360 KEY=DOGS
>N
P=0002 KEY=34
P=0002 KEY=LAW
>RE
ENTER KEY
>CARS
P=0359  NM=103
P=0359  NM=CARS
ENTER COMMAND
>SC
P=0359 KEY=CARS
P=0359 KEY=103
>N
P=0309 KEY=CELTIC PEOPLE
P=0309 KEY=(=6)
>P
P=0359 KEY=CARS
P=0359 KEY=103
>P
P=0358 KEY=CARRIER BAGS
P=0358 KEY=102
>P
P=0081 KEY=CARACAS CONFERENCE
P=0081 KEY=341.225.06 "1974"
```

```
>DELETE
ENTER KEY
>103
P=0359  NM=103
P=0359  NM=CARS
DELETE OR SCAN
>DE
ENTER COMMAND
>DE
ENTER KEY
>102
P=0358  NM=102
P=0358  NM=CARRIER BAGS
DELETE OR SCAN
>SC
P=0358 KEY=102
P=0358 KEY=CARRIER BAGS
>N
P=0357 KEY=102
P=0357 KEY=BAGS
>DE
ENTER COMMAND
>MODE
ENTER MODE, UDC OR AS
>UDC
ENTER COMMAND
>RE
ENTER KEY
>340.77
P=5567  NM=340.77
ENTER COMMAND
>@
END OF EXECUTION
>@FIN


  RUNID: ROSTFF   ACCT: SCS              PROJECT: BSTAFF
  TIME:     TOTAL: 00:03:25.919      CBSUPS:  030669382
            CPU:   00:00:31.050      I/O:  00:01:51.890
            CC/ER: 00:01:02.979      WAIT: 00:18:23.833

  SUAS USED:  $    11.32   SUAS REMAINING:  $  9846.07
  IMAGES  READ:   112       PAGES:  40
  START:    10:21:34 DEC 08,1977    FIN:  10:43:20 DEC 08,1977
*TERMINAL INACTIVE*
>@@TERM
```

A P P E N D I X 3

A design language for the definition of a retrieval system interface for casual users of a relational database

by

P.G. Thomas and B.E. Staff

Faculty of Mathematics
The Open University

August 1978

A design language for the definition of a retrieval system interface for casual users of a relational database

Abstract

A design language is described that enables the information analyst to produce a tailor-made interface for the casual user of a relational database system. The ideas are extended to show how "ad hoc" retrieval systems can be constructed from the building blocks incorportated in the relational model.

Introduction

Databases, like computers in general, are used at a number of levels according to one's experience and requirements. A relational database designed around the relational algebra will most commonly be manipulated by means of low level commands such as SELECT, PROJECT and JOIN -[1,2,3]- which, in that they require a little knowledge of database technique to be employed effectively, do not constitute the sort of interaction that the most casual of users should be expected to undertake. One method of getting round this is to embed the low level commands in a host language (eg. PL1 [4]) and then write programs in that host language which present a more hospitable face to the user. Also, direct manipulation languages have been developed that allow complex relational operations to be notated with greater facility -[5,6,7,8,9]. In this paper however, we discuss a method by which the information analyst can build a retrieval interface of any degree of simplicity from:-

(i) the basic set of relational operations - SELECT, PROJECT and JOIN,

(ii) arithmetic and conditional operators -eg. +, - , >, etc.,

(iii) a short library of declarative and command statements - eg. ADDTUPLE, GOTO, IF etc.

The procedures written by the analyst, which will be tailor-made to the user's needs, are run interpretively, thus by-passing lengthy compilations and enabling the analyst to write and test his interface at a terminal.

Computer Details

All of the programs described in this paper were written in BASIC to run on the Hewlett Packard 2000F computer belonging to the Open University's Student Computing Service. A maximum permitted program size of 10k words requires that programs be continually swapped in and out of core, but efficiency is maintained and with it the hope that conversion to a microprocessing system might eventually be possible.

The Relational Database

This specially constructed relational database system makes use of B^* trees -[10,11]- for the organization of key domains. In addition to the well established advantages accruing to the use of B trees (maintenance of balance, localized updating etc.) is added, in the case of the B^* tree, the ability to perform a rapid scan across the leaf pages, which was found particularly useful when JOINING relations -[12].

The Design Language

The central requirement of the design language for the information analyst to use, is for a means of expressing complex series of relational operations with the greatest facility. It would have been possible to demand the use of relational operation commands in their literal form, eg:-

```
SELECT from RELATION EMPLOYEE SUCH THAT ENAME = "SMITH"
                    YIELDING RELATION EMP2
```

PROJECT RELATION EMP2 ON ENAME and ENUM

YIELDING RELATION EMP3

It was decided in preference however, that relational operations should be expressed in a lightly notated form. The advantages of this choice are brevity of expression and the possibility of defining a complex series of operations in a single statement, whilst the disadvantage for the analyst using the language, is be the need to come to terms with the simple notation described below:-

Input Relations: V(A,B) and W(D,C)

Output Relations: X(A,B), Y(A) and Z(A,B,C)

SELECTION from V such that A = n is notated thus:-

$$X(A,B) = V(A=n\ ,B)$$

PROJECTION of V on domain A is notated thus:

$$Y(A) = V(A)$$

where the missing subscript B means that this domain is not to be projected.

an equi-JOIN (over one domain) of V and W on domains B and D is notated thus:-

$$Z(A,B,C) = V(A,B)*(B!=D)*W(D,C)$$

where the ! symbol denotes the name to be given to the domain resulting from the JOIN.

When operations are compounded, the notation can be read from left to right, e.g:-

$$Z(A,B,C) = (V(A,B)*(B!=D)*W(D,C))(B>n\ ,C=\ 2)*(B!>B)*X(A,B).$$

(1) (2) (3)

This expression could be expressed literally as follows, where the

TEMP relations mentioned below are all taken care of by the system, and the analyst only has to provide an adequate Left Hand Side:-

(1) JOIN(=) relations V and W on domains B and D → TEMP1(A,B,C)

(2) (i) PROJECT TEMP1 on domains B & C → TEMP2(B,C)
(i.e. mention of domain A is not contained in brackets (2))

(ii) SELECT from TEMP2 such that B>n & C= 2 → TEMP3(B,C)

(3) JOIN(>) TEMP3 with X on domains B & B → Z(A,B,C)

Although it is in no sense a relational operation, the analyst can also specify that arithmetic operations be performed over whole domains eg:-

X(A,B) = V (A+29, B*26.7/395.8)

The language requires that certain declarations be made before relations can be manipulated.

The REL statement concerns relation declarations in respect of domain names, types and upper and lower bounds. The OPEN statement directs that an existing relation be retrieved from backing-store, and checked against the declaration of its format made in the corresponding REL statement. Relations declared - by REL - but <u>not</u> OPENed, are assumed to be <u>new</u> relations that will be constructed from operations on existing relations. Such a new relation must, at some stage, appear on the Left Hand Side of a relational expression, and it can then be the subject of a SAVE command, causing it to be made permanent on the backing store. In general, the SAVE command causes a relation to be copied to backing-store, overwriting the version of the same relation (if any) that previously existed there.

The other commands in the design-language permit conditional and

and unconditional branching (IF & GOTO) and simple manipulation of arithmetic variables by way of PLUSAB (plus and becomes) and MINAB (minus and becomes) statements. Provision for more complex operations is satisfied by allowing the analyst to incorporate his own BASIC subprograms to perform any desired function and, if necessary, return values to the design language runstream.

In this way, procedures can be written for relation manipulation (which might typically involve payroll computation, inventory report generation etc.) that would be produced by an analyst familiar with the principles of a relational database, and activated by a casual user simply typing the procedures name. Any additional information required from the user, is solicited as and when required (values for constants etc.) and may be preceded by any number of explanatory messages.

The facility also exists for the addition of tuples to a relation by solicitation from the casual user (ADDTUPLE command). To assist in this process, the analyst can affix an explanatory message to each domain to which a value is to be ascribed.

For example, given a relation EMPLOYEE (Enum, Ename), which the user is required to update by the addition of new tuples, the analyst includes the following statements in his procedural runstream:-

```
ADDTUPLE EMPLOYEE
ENUM = ENTER EMPLOYEE's NUMBER
ENAME = AND NAME, EG. SMITH, A.B.
END$
```

When the procedure containing these statements is executed, the casual user inter-acts as follows (the user's response is underlined):-

```
TYPE YES IF YOU WANT HELP:-YES
 2      DOMAINS ARE REQUIRED OF THE FOLLOWING FORMAT:-
 1      TYPE=REAL; LB= 1     UB= 100
 2      TYPE=CHAR; MAX WIDTH= 10
FIRST TIME ROUND YOU MUST GIVE A VALUE TO EVERY DOMAIN
AFTER WHICH, IF A VALUE FOR A DOMAIN IS THE SAME AS THAT IN
THE PREVIOUS TUPLE YOU CAN SIMPLY PRESS C/R. BEFORE EACH
NEW TUPLE IS SOLICITED YOU WILL BE ABLE TO STOP THE
PROCESS BY TYPING END

TYPE END TO STOP ADDING:- ___ .....return (c/r) key was pressed
 1     ENTER EMPLOYEE'S NUMBER?45
 2     AND NAME, EG. SMITH,A.B.?JONES,D.G.
CALL ADDTUP....WITH WFILE VALUES:-
 45    JONES,D.G.

TYPE END TO STOP ADDING:-___ .........c/r
 1     ENTER EMPLOYEE'S NUMBER?96
 2     AND NAME, EG. SMITH,A.B.?HUME,D.
CALL ADDTUP....WITH WFILE VALUES:-
 96    HUME,D.

TYPE END TO STOP ADDING:-____ .........c/r
 1     ENTER EMPLOYEE'S NUMBER?68
 2     AND NAME, EG. SMITH,A.B.?____ c/r  HUME,D.    ASSUMED

CALL ADDTUP....WITH WFILE VALUES:-
 68    HUME,D.

TYPE END TO STOP ADDING:-___ .........c/r
 1     ENTER EMPLOYEE'S NUMBER?46
 2     AND NAME, EG. SMITH,A.B.?BURNS,K.D.
CALL ADDTUP....WITH WFILE VALUES:-
 46    BURNS,K.D.

TYPE END TO STOP ADDING:-END

DONE
```

The interface with the casual user is, therefore, left entirely in the hands of a benign mediator who , knowing the user's ability, is in the best position to provide the appropriate level of condescension. Any interaction required of the user can be made as explicit as the analyst sees fit.

In the following examples, the salient features of the design language in which the information analyst deals are, it is hoped, elucidated. Anything following a semi-colon is for comment only.

EXAMPLE 1 : A SELECT is performed on input relation PART(Pnum, Pname) according to parameters supplied by the user (solicited and vetted in the CALLED routine MYSOL) giving output relation PT(Pnum, Pname).

```
1.  REL PART(Pnum/I(1:99), Pname/C(10))   ; I=Integer, C=Character
2.  REL PT(Pnum/I(1:99), Pname/C(10))
3.  OPEN PART;  retrieve relation from backing store
4.  CALL MYSOL; routine supplied by the analyst
5.  RETURN VAR1, VAR2
6.  PT(Pnum, Pname) = PART(Pnum = VAR1, Pname = VAR2)
7.  PRINT PT
8.  SAVE PT;  save new relation on backing store
9.  END.
```

EXAMPLE 2 : A value, VAR2, is computed in subroutine MYCOMP as a function of variable VAR1, which is varied from 1 to 3 in the design runstream. VAR2 is then used as a parameter in a relational operation.

```
1.   REL WAGE(Ename/C(15), Sum /I(1000:5000), Tax /R(0.15:96.54))
2.   REL DEPT(Dname/C(5), Ename/C(15))
3.   REL DSAL(Dname/C(5), Ename/C(15), Sum /I(1000:5000), Tax /R(0.15:96.54))
4.   OPEN WAGE
5.   OPEN DEPT
6.   PLUSAB VAR1;  first mention of VAR1 initializes it to zero.
7.   PLUSAB VAR1
8.   IF VAR1 > 3 THEN 15
9.   CALL MYCOMP (VAR1);  routine supplied by the analyst
10.  RETURN VAR2
11.  DSAL (Dname, Ename, Sum, Tax) = Dept(Dname = "LUTON", Ename)
12.  *(Ename!=Ename)*WAGE(Ename, Sum>2000, Tax*VAR2).
13.  PRINT DSAL
14.  GO TO 7
15.  END
```

Lines 11 and 12 tabulate the salary details of everyone in the "LUTON" department having a salary of more than £2000 pa, with the adjusted tax figure.

[Note - since the relation DSAL is not SAVED, it is assumed to be temporary and is discarded at the end of the run.]

EXAMPLE 3 : New tuples are added to the EMP relation, and the salary table is recomputed.

```
1.   REL EMP(Ename/C(10), Etype/I(1:100)), ES(Etype/I(1:100), Esal/I(1:5000))
2.   REL SALARY(Ename/C(10), Esal/I(1:5000))
3.   OPEN EMP
4.   OPEN ES
5.   CALL ASKUSER;  routine supplied by the analyst to provide starting
                    information.
6.   RETURN YESNO, PASSWORD
7.   IF YESNO="NO" then 17
8.   IF PASSWORD≠ "X104" then 17
9.   ADDTUPLE EMP
10.  Ename = "Enter new employee's name"
11.  Etype = "And his earnings type"
12.  END$
13.  SALARY(Ename, Esal) = (EMP(Ename,Etype)*(Etype!=Etype)*
14.  ES(Etype,Esal)) (Ename,Esal).
15.  PRINT SALARY
16.  GO TO 5
17.  SAVE EMP
18.  END
```

In line 17, the new version of relation EMP (with added tuples) is saved on the backing-store, overwriting the previous version.

An application : The modular design of information retrieval systems

A particular application being confronted at the same time as the work hitherto described, was a retrieval system for use in a film library organized according to a long established classification scheme (the Universal Decimal Classification or UDC). In order to maximize methods of access to film items, it was required that:-

(i) various files be made available for independent or concertive use, and that

(ii) a highly specific retrieval system be developed for the interrogation of the 400,000 UDC numbers assigned to reels of film.

A relational database would immediately suggest itself in respect of point (i). The standard relational operations however, would not have constituted anything approaching the levels of specificity and efficiency required of point (ii), and so a tailor-made system was designed and built to fulfil this need.

This step, which on the face of it might seem to have ruled out the parallel use of a relational database, in fact would do nothing of the sort. If one looks upon the specially written UDC retrieval system as a particularly complex form of the SELECT operation, then it can be appreciated that any specific retrieval system that takes a relation as input and produces a relation as output, can be considered as an additional relational operation which in no way transgresses the integrity of the database. Extending this idea, it would be possible to have any number of highly specific retrieval methods ("retrieval modules") available to the information analyst, to compliment the basic relational operations:-

Relation $A(d_1, d_2 \ldots d_n)$ cardinality = N1

↓

Retrieval Module

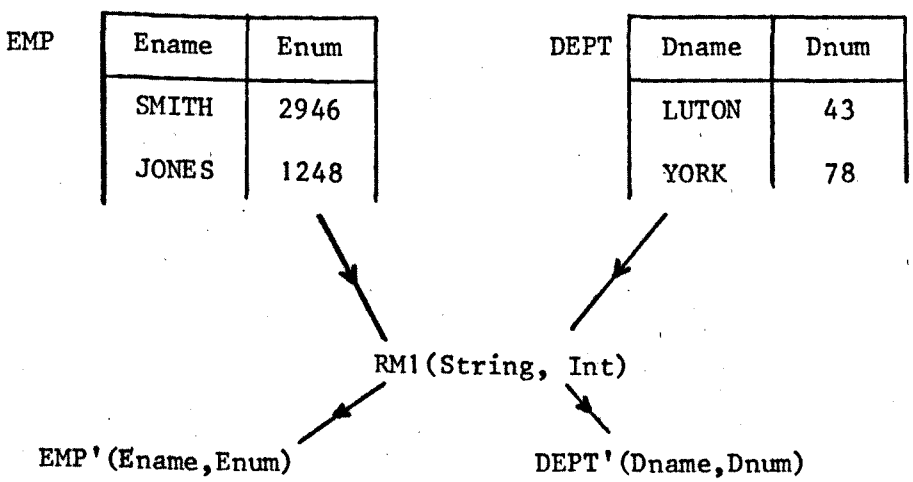
RM1

↓

Relation $B(d_1, d_2 \ldots d_n)$ cardinality = N2

where, to be useful, N2<N1 - RM1 constituting a partitioning operation.

The retrieval modules would, like relational operations, be distinct from the data structures (relations) upon which they act, and would be defined purely in terms of the types of fields (domains) to which their complex selection processes can be applied. For example, RM1 operates on two fields, one integer and one string, and it could therefore be used to refine any relation having a domain of each of these types, eg:-



The retrieval modules, which the analyst would be able to write for his own purposes as long as they satisfied the relation-in relation-out condition, could be of any degree of complexity. In the film library UDC retrieval module for example, string fields are examined for sub-strings occurring within other substrings - an operation that the straightforward relational SELECT would not be able to handle.

The present design language can accommodate the introduction of analyst written retrieval modules by means of the subroutine CALL facility. In time however, it is expected to replace this by incorporating modular retrieval in the relational notation itself, eg:-

(1) EMP'(Ename,Enum) = RM1[EMP(Ename,Enum)]

i.e. operate on relation EMP using retrieval module RM1 to produce relation on EMP'.

(2) NAMSAL(Ename,Esal) = (RM1[EMP(Ename,Enum)]*
(Enum!=Enum)*RM2[SAL(Enum,Esal)])(Ename,Esal).

i.e. operate on EMP using RM1 → TEMP1 (a relation created by the system)
operate on SAL using RM2 → TEMP2 (a relation created by the system)
JOIN TEMP1 and TEMP2 → TEMP3(Ename, Enum, Esal)
PROJECT TEMP3 on Ename and Esal → NAMSAL

Summary

In fact, it was from the independence (normalization) of data stored in relations that the idea of modular retrieval system design sprang, in the belief that a set of independent retrieval modules was the obvious complement to a set of independent data structures. Why should not the much acclaimed "application independence" of databases in general be applied to retrieval procedures as much as to data structures? Using the design language outlined earlier, the analyst would then be able to construct "ad hoc" retrieval systems from modular data and procedural building blocks to suit his users.

References

(1) CODD, E.F. "A relational model of data for large shared data banks". Comm ACM 13 (6), June 1970.

(2) CODD, E.F. "Relational completeness of data sublanguages". Courant Comp. Sci. Symposia 6, Data Base Systems, Prentice-Hall, New York, May 1971.

(3) CODD, E.F. "Normalized Data Base Structure : A Brief Tutorial" Proc. 1971 ACM Sigfidet Workshop on Data Description Access and Control, San Diego, Nov. 1971.

(4) MYLOPOULOS, J. et al "A multi-level relational system". National Computing Conference 1975, ps 403-408.

(5) ASTRAHAN, M.M. and CHAMBERLIN, D.D. "Implementation of a structured English query language" Comm. ACM 8, (10), Oct, 1975.

(6) CHAMBERLIN, D.D. et al "SEQUEL 2 : A unified approach to data definition, manipulation and control". IBM J. Res. Dev., Nov. 1976.

(7) HELD, G.D. et al "INGRES - a relational database system" National Computing Conference 1975, ps 409-416.

(8) ZLOOF, M.M. "Query by example" National Computing Conference 1975, ps 431-437.

(9) MACLEOD, I.A. "Towards an information retrieval language based on a relational view of data" Info. Proc. and Man. 13, 1977

(10) BAYER, R. and McCREIGHT, E. "Organization and maintenance of large ordered indexes".
Acta Information 1, 1972.

(11) KNUTH, D. "The Art of Computer Programming - Volume 3 - Sorting and Searching"
Addison Wesley, Reading, Mass. 1973, ps 473-479.

(12) THOMAS, P.G.
Open University Computing Group. Internal Report.